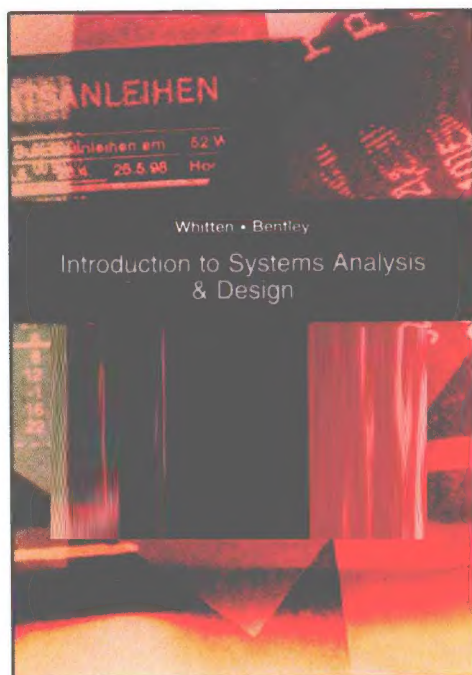


系统分析与设计导论

(美) **Jeffrey L. Whitten Lonnie D. Bentley** 著 肖刚 孙慧 等译
普度大学 普度大学

Introduction to Systems Analysis and Design



系统分析与设计导论

Introduction to Systems Analysis and Design

本书是经典教材《系统分析与设计方法》的简明版本，既保留了经典教材内容全面的特色，又对高级主题进行了适当的精减，更强调系统概念，使之更加适用于导论课教学要求。

全书详细阐述面向对象系统分析和设计技术。作者通过融入基于UML的面向对象分析和设计技术，对现代概念、工具、技术以及应用等各方面内容进行了很好的平衡。本书提供了市场上可用的、丰富的系统分析和设计的实例。

作者简介

Jeffrey L. Whitten 美国普度大学计算机技术系主任兼教授，曾两次荣获James G. Dwyer最佳教师奖。自1984年任教授后，他开始编著《系统分析与设计方法》一书，目前已经出版到第7版。该书长期位于同类书销售排行榜第1名，被700多所学校采纳作为教材。Whitten教授是多个学术组织的活跃成员，其中包括：信息技术专业学会(ATP)、信息系统学会(AIS)、计算机学会(ACM)、信息管理协会(SIM)等。

Lonnie D. Bentley 美国普度大学计算机技术系教授，主要教学和研究领域包括：系统分析和设计、企业应用系统、业务过程重构、计算机辅助软件工程(CASE)、快速应用开发(RAD)和图形用户界面设计。



系统分析与设计导论 (英文版)
书号: 978-7-111-35278-5
定价: 79.00元



系统分析与设计方法 (原书第7版)
书号: 978-7-111-20551-7
定价: 59.00元

客服热线: (010) 88378991, 88361066
购书热线: (010) 68326294, 88379649, 68995259
投稿热线: (010) 88379604
读者信箱: hzsj@hzbook.com

华章网站 <http://www.hzbook.com>



Mc
Graw
Hill

Education

<http://www.mheducation.com>

网上购书: www.china-pub.com

封面设计: 陈敬

ISBN 978-7-111-36386-6

ISBN 978-7-111-36386-6



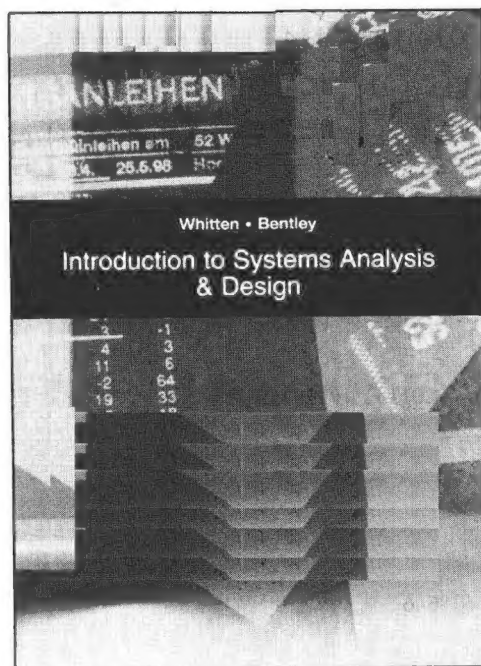
定价: 49.00元

计 算 机 科 学 丛 书

系统分析与设计导论

(美) Jeffrey L. Whitten Lonnie D. Bentley 著 肖刚 孙慧 等译
普度大学 普度大学

Introduction to Systems Analysis and Design



机械工业出版社
China Machine Press

本书是经典教材《系统分析与设计方法》的简明版本,综合而全面地介绍计算机系统分析与设计方法,集学术和实践于一体,既保留了经典教材内容全面的特色,又对高级主题进行了适当的精减,更强调系统概念,使之更加适用于系统分析与设计导论课的教学要求。

全书内容翔实、脉络清晰、分析透彻、例证充分,适合作为高等院校计算机、信息系统及商科相关专业信息系统开发或系统分析与设计导论课程的教材或参考书,也可供专业技术人员参考。

Jeffrey L. Whitten, Lonnie D. Bentley: Introduction to Systems Analysis and Design (ISBN 0-07-340294-X). Copyright © 2008 by The McGraw-Hill Companies, Inc..

All Rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including without limitation photocopying, recording, taping, or any database, information or retrieval system, without the prior written permission of the publisher.

This authorized Chinese translation edition is jointly published by McGraw-Hill Education (Asia) and China Machine Press. This edition is authorized for sale in the People's Republic of China only, excluding Hong Kong, Macao SAR and Taiwan.

Copyright © 2012 by McGraw-Hill Education (Asia), a division of the Singapore Branch of The McGraw-Hill Companies, Inc. and China Machine Press.

版权所有。未经出版人事先书面许可,对本出版物的任何部分不得以任何方式或途径复制或传播,包括但不限于复印、录制、录音,或通过任何数据库、信息或可检索的系统。

本授权中文简体字翻译版由麦格劳-希尔(亚洲)教育出版公司和机械工业出版社合作出版。此版本经授权仅限在中华人民共和国境内(不包括香港特别行政区、澳门特别行政区和台湾)销售。

版权© 2012 由麦格劳-希尔(亚洲)教育出版公司与机械工业出版社所有。

本书封面贴有 McGraw-Hill 公司防伪标签,无标签者不得销售。

封底无防伪标均为盗版

版权所有,侵权必究

本书法律顾问 北京市展达律师事务所

本书版权登记号: 图字: 01-2007-3114

图书在版编目(CIP)数据

系统分析与设计导论 / (美)惠滕(Whitten, J. L.), (美)本特利(Bentley, L. D.)著;肖刚,孙慧译. —北京:机械工业出版社,2012.1

(计算机科学丛书)

书名原文: Introduction to Systems Analysis and Design

ISBN 978-7-111-36386-6

I. 系… II. ①惠… ②本… ③肖… ④孙… III. ①信息系统—系统分析—教材 ②信息系统—系统设计—教材 IV. G202

中国版本图书馆CIP数据核字(2011)第229159号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑:姚蕾

北京京北印刷有限公司印刷

2012年1月第1版第1次印刷

185mm×260mm·27.25印张

标准书号: ISBN 978-7-111-36386-6

定价: 49.00元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

客服热线: (010) 88378991; 88361066

购书热线: (010) 68326294; 88379649; 68995259

投稿热线: (010) 88379604

读者信箱: hzjsj@hzbook.com

出版者的话
前言
致谢

1.5.2	系统分析	16
1.5.3	系统设计	17
1.5.4	系统实现	18
1.5.5	系统支持和持续改进	18

第一部分 系统开发项目环境

第1章 系统分析和设计方法的

环境	2
1.1 产品——信息系统	4
1.2 参与者——系统关联人员	4
1.2.1 系统所有者	4
1.2.2 系统用户	5
1.2.3 系统设计人员	6
1.2.4 系统构造人员	6
1.2.5 系统分析员	7
1.2.6 外部服务提供者	7
1.2.7 项目经理	7
1.3 现代信息系统的业务驱动力	7
1.3.1 经济全球化	8
1.3.2 电子商务和电子业务	8
1.3.3 安全和隐私	10
1.3.4 协作与合伙经营	10
1.3.5 知识产权管理	10
1.3.6 持续改进和全面质量管理	10
1.3.7 业务过程重构	10
1.4 信息系统的技术推动力	11
1.4.1 网络和因特网	11
1.4.2 移动和无线技术	12
1.4.3 对象技术	12
1.4.4 协作技术	13
1.4.5 企业应用软件	13
1.5 过程——系统开发过程	16
1.5.1 系统启动	16

第2章 信息系统开发

2.1 系统开发过程	24
2.1.1 能力成熟度模型	24
2.1.2 系统生命周期和系统开发 方法	25
2.1.3 系统开发基本原理	26
2.2 系统开发过程	28
2.2.1 项目确定	28
2.2.2 项目开发阶段	30
2.2.3 跨生命周期活动	36
2.2.4 顺序开发和迭代开发	37
2.3 选择开发路线和策略	37
2.3.1 模型驱动开发策略	41
2.3.2 快速应用开发策略	43
2.3.3 商用应用软件包实现策略	45
2.3.4 混合策略	47
2.3.5 系统维护	47
2.4 自动化工具和技术	48
2.4.1 计算机辅助系统工程	48
2.4.2 应用开发环境	51
2.4.3 过程和项目管理器	52

第3章 项目管理

3.1 什么是项目管理	56
3.1.1 项目失败的原因	57
3.1.2 项目管理知识体系	58
3.2 项目管理生命周期	62
3.2.1 活动 I——协商范围	63

3.2.2	活动2——确定任务	64
3.2.3	活动3——估计任务工期	64
3.2.4	活动4——说明任务之间的依 赖关系	65
3.2.5	活动5——分配资源	66
3.2.6	活动6——指导团队工作	69
3.2.7	活动7——监督和控制进展	70
3.2.8	活动8——评估项目结果和 经验	75

第二部分 系统分析方法

第4章 系统分析

4.1	什么是系统分析	83
4.2	系统分析方法	84
4.2.1	模型驱动分析法	84
4.2.2	加速系统分析法	86
4.2.3	需求获取法	87
4.2.4	业务过程重构法	88
4.2.5	系统分析策略	88
4.3	范围定义阶段	88
4.3.1	任务1.1——列出问题和 机会	89
4.3.2	任务1.2——协商项目的 初步范围	91
4.3.3	任务1.3——评估项目价值	92
4.3.4	任务1.4——计划项目进度表和 预算	92
4.3.5	任务1.5——汇报项目计划	92
4.4	问题分析阶段	93
4.4.1	任务2.1——研究问题领域	93
4.4.2	任务2.2——分析问题和机会	96
4.4.3	任务2.3——分析业务过程	96
4.4.4	任务2.4——制定系统改进 目标	97
4.4.5	任务2.5——修改项目计划	98
4.4.6	任务2.6——汇报调查结果和 建议	98
4.5	需求分析阶段	99

4.5.1	任务3.1——定义需求	100
4.5.2	任务3.2——排列需求的优 先次序	100
4.5.3	任务3.3——修改项目计划	100
4.5.4	任务3.4——交流需求陈述	101
4.5.5	持续不断的需求管理	101
4.6	逻辑设计阶段	101
4.6.1	任务4.1a——结构化功能 需求	101
4.6.2	任务4.1b——建立功能需求 的原型(可选)	101
4.6.3	任务4.2——验证功能需求	102
4.6.4	任务4.3——定义验收测 试用例	102
4.7	决策分析阶段	102
4.7.1	任务5.1——确定候选方案	103
4.7.2	任务5.2——分析候选方案	105
4.7.3	任务5.3——比较候选方案	105
4.7.4	任务5.4——修改项目计划	106
4.7.5	任务5.5——推荐一种系统 方案	107

第5章 需求获取的调查研究

技术

5.1	需求获取简介	111
5.2	需求获取过程	112
5.2.1	发现和分析问题	112
5.2.2	获取需求	113
5.2.3	归档和分析需求	113
5.2.4	需求管理	114
5.3	调查研究技术	114
5.3.1	对现有文档、表和文件进行 抽样	114
5.3.2	调研和实地访问	116
5.3.3	观察工作环境	116
5.3.4	调查表	117
5.3.5	面谈	119
5.3.6	如何进行面谈	119
5.3.7	获取原型	123

5.3.8 联合需求计划	123	7.5.3 规范化举例	169
5.4 调查研究策略	126	7.6 将数据需求映射到地点	175
第6章 使用用例建模系统需求 ...	130	第8章 过程建模	178
6.1 用例建模简介	131	8.1 过程建模简介	179
6.2 用例建模的系统概念	131	8.2 过程建模的系统概念	179
6.2.1 用例	132	8.2.1 外部代理	179
6.2.2 参与者	132	8.2.2 数据存储	181
6.2.3 关系	133	8.2.3 过程概念	182
6.3 需求用例建模过程	134	8.2.4 数据流	185
6.3.1 第1步: 确定业务参与者	134	8.3 逻辑过程建模的过程	192
6.3.2 第2步: 确定业务需求用例 ...	135	8.4 如何构造过程模型	193
6.3.3 第3步: 构造用例模型图	136	8.4.1 上下文数据流图	194
6.3.4 第4步: 记录业务需求用例 描述	137	8.4.2 功能分解图	194
6.4 用例和项目管理	139	8.4.3 事件响应或用例清单	196
6.4.1 分级和评估用例	141	8.4.4 事件分解图	197
6.4.2 确定用例依赖关系	141	8.4.5 事件图	198
第7章 数据建模和分析	145	8.4.6 系统图	200
7.1 数据建模简介	146	8.4.7 基本图	200
7.2 数据建模的系统概念	147	8.4.8 完成规格说明	203
7.2.1 实体	147	第9章 使用UML进行面向对象 分析和建模	210
7.2.2 属性	148	9.1 面向对象分析简介	211
7.2.3 关系	150	9.2 对象建模的系统概念	211
7.3 逻辑数据建模过程	158	9.2.1 对象、属性、方法和封装	211
7.3.1 战略数据建模	158	9.2.2 类、泛化和特化	212
7.3.2 系统分析期间的数据建模	158	9.2.3 对象/类关系	214
7.3.3 对系统设计的考虑	159	9.2.4 消息和消息发送	216
7.3.4 数据建模的自动化工具	159	9.2.5 多态性	216
7.4 如何构造数据模型	160	9.3 UML模型图	217
7.4.1 获取实体	160	9.4 对象建模过程	218
7.4.2 上下文数据模型	161	9.4.1 建模系统的功能性描述	218
7.4.3 基于键的数据模型	163	9.4.2 构造分析用例模型	218
7.4.4 泛化层次体系	165	9.4.3 建模用例活动	224
7.4.5 具有完整属性的数据模型	165	9.4.4 构造活动图指南	226
7.5 分析数据模型	167	9.4.5 绘制系统顺序图	227
7.5.1 好的数据模型的标准	168	9.4.6 构造系统顺序图指南	228
7.5.2 数据分析	168	9.4.7 发现和确定业务对象	228

9.4.8 组织对象并确定其关系	230
------------------------	-----

第10章 可行性分析和系统方案

建议	238
10.1 可行性分析和系统方案建议	238
10.1.1 可行性分析——逐步投入法	239
10.1.2 系统分析——范围定义阶段的检查点	239
10.1.3 系统分析——问题分析阶段的检查点	239
10.1.4 系统设计——决策分析阶段的检查点	239
10.2 可行性的6个准则	241
10.2.1 运行可行性	241
10.2.2 技术可行性	241
10.2.3 进度可行性	241
10.2.4 经济可行性	242
10.3 成本效益分析技术	242
10.3.1 系统将花费多少	242
10.3.2 系统将提供什么收益	243
10.3.3 建议的系统合算吗	244
10.4 候选系统的可行性分析	247
10.4.1 候选系统矩阵	247
10.4.2 可行性分析矩阵	249
10.5 系统方案建议	250
10.5.1 书面报告	251
10.5.2 正式汇报	252

第三部分 系统设计方法

第11章 系统设计

11.1 什么是系统设计	259
11.2 系统设计方法	259
11.2.1 模型驱动方法	260
11.2.2 快速应用开发	263
11.2.3 系统设计策略	263
11.3 系统设计之内部开发——“构造”方案	263

11.3.1 任务5.1——设计应用架构 ...	263
11.3.2 任务5.2——设计系统数据库	266
11.3.3 任务5.3——设计系统接口 ...	268
11.3.4 任务5.4——打包设计说明 ...	269
11.3.5 任务5.5——修改项目计划 ...	269
11.4 系统设计之集成商用软件——“购买”方案	269
11.4.1 任务4.1——研究技术评价准则和选项	271
11.4.2 任务4.2——向供应商征求建议（或报价）	272
11.4.3 任务5A.1——验证供应商的声明和性能	272
11.4.4 任务5A.2——评价和分级供应商建议	273
11.4.5 任务5A.3——签订合同并听取供应商汇报	273
11.4.6 购买决定对剩余生命周期阶段的影响	274

第12章 应用架构和建模

12.1 应用架构	279
12.2 物理数据流图	279
12.2.1 物理过程	280
12.2.2 物理数据流	282
12.2.3 物理外部代理	284
12.2.4 物理数据存储	284
12.3 信息技术架构	285
12.3.1 分布式系统	285
12.3.2 数据架构——分布式关系数据库	292
12.3.3 接口架构——输入、输出和中间件	293
12.3.4 过程架构——软件开发环境 ...	297
12.4 建模信息系统应用架构	299
12.4.1 绘制物理数据流图	299
12.4.2 网络架构	299
12.4.3 数据分布和技术确定	300

12.4.4 过程分布和技术确定	301	15.1.3 输入设计的系统用户问题	348
12.4.5 人/机边界	302	15.1.4 内部控制——输入数据的 编辑	349
第 13 章 数据库设计	307	15.2 输入设计的 GUI 控件	350
13.1 系统分析员的数据库概念	308	15.2.1 常用 GUI 输入控件	351
13.1.1 字段	308	15.2.2 高级输入控件	354
13.1.2 记录	308	15.3 如何设计和原型化输入	355
13.1.3 文件和表	309	15.3.1 输入设计和原型化的自动化 工具	355
13.1.4 数据库	309	15.3.2 输入设计过程	355
13.2 数据库设计的前置条件—— 规范化	314	15.3.3 基于 Web 的输入和电子业务	359
13.3 现代数据库设计	315	第 16 章 用户界面设计	363
13.3.1 数据库设计的目标和前置 条件	315	16.1 用户界面设计概念和指南	363
13.3.2 数据库模式	315	16.1.1 计算机用户的类型	364
13.3.3 数据完整性和访问完整性	319	16.1.2 人的因素	364
13.3.4 角色	320	16.1.3 人类工程学指南	364
13.3.5 数据库分布和复制	321	16.1.4 对话语气和词汇	365
13.3.6 数据库原型	321	16.2 用户界面技术	366
13.3.7 规划数据库容量	321	16.2.1 操作系统和 Web 浏览器	366
13.3.8 数据库结构生成	322	16.2.2 显示器	366
第 14 章 输出设计和原型化	325	16.2.3 键盘和指点设备	367
14.1 输出设计概念和指南	325	16.3 图形用户界面风格	367
14.1.1 输出的分布和观众	326	16.3.1 窗口和框	367
14.1.2 输出的实现方法	327	16.3.2 菜单驱动的界面	368
14.2 如何设计和原型化输出	331	16.3.3 指令驱动的界面	373
14.2.1 用于输出设计和原型化的 自动化工具	331	16.3.4 提问—回答对话	374
14.2.2 输出设计指南	332	16.3.5 用户界面设计的特殊考虑	375
14.2.3 输出设计过程	333	16.4 如何设计用户界面	378
14.2.4 基于 Web 的输出和电子 业务	339	16.4.1 用于用户界面设计和原型化 的自动化工具	378
第 15 章 输入设计和原型化	344	16.4.2 用户界面设计过程	378
15.1 输入设计概念和指南	344	第 17 章 使用 UML 进行面向对象 设计和建模	385
15.1.1 数据收集、数据录入和数据 处理	345	17.1 设计面向对象系统	385
15.1.2 输入方法和实现	346	17.1.1 实体类	386
		17.1.2 接口类	386
		17.1.3 控制类	386

17.1.4	持续类	386
17.1.5	系统类	387
17.1.6	设计关系	387
17.1.7	属性和方法可见性	387
17.1.8	对象责任	387
17.2	面向对象设计过程	388
17.2.1	精炼用例模型	388
17.2.2	建模支持用例情境的类 交互、行为和状态	389
17.2.3	修改对象模型以反映实现 环境	396

第四部分 系统分析和设计 完成后的工作

第 18 章	系统构造和实现	402
18.1	什么是系统构造和实现	402

18.2	构造阶段	402
18.2.1	任务 6.1——构建和测试网络 (如果需要)	402
18.2.2	任务 6.2——构建和测试 数据库	405
18.2.3	任务 6.3——安装和测试新 软件包 (如果需要)	405
18.2.4	任务 6.4——编写和测试新 程序	405
18.3	实现阶段	406
18.3.1	任务 7.1——进行系统测试	406
18.3.2	任务 7.2——准备转换计划	406
18.3.3	任务 7.3——安装数据库	408
18.3.4	任务 7.4——培训用户	408
18.3.5	任务 7.5——转换到新系统	409

词汇表	412
-----------	-----

系统开发项目环境

这是一本关于如何实践信息系统开发方法的书。所有的企业和组织都会开发信息系统，而且可以肯定，你将在某些系统的系统分析和设计中扮演某个角色，也许是作为系统的客户或用户，也许是作为那些系统的开发者。系统分析和设计技术涉及业务问题的解决和计算机应用程序的开发，本书中的方法也可以应用于更广泛的问题领域，而不仅仅是计算机领域。

在开始学习之前，我们假定读者已经完成一门计算机信息系统导论课程的学习，并且许多读者也学过一门或几门程序设计课程（例如 Access、Java、C/C++ 或 Visual Basic）。学习这些课程是有益的，因为系统分析和设计需要这些基础知识并综合运用了这些知识。如果读者没有学过这些课程，也不必担心，我们会复习系统分析和设计所需的基本原理。

本书第一部分集中于系统分析和设计的整体描述。在学习具体的活动、工作、技巧、方法和技术之前，首先需要理解这个整体描述。但当你阅读系统分析和设计环境这一部分时，我们介绍的许多思想、工具和技巧将在本书后面才详细讲述。请记住这一点！

系统开发并不神奇。这里既没有成功的秘诀，也没有完美的工具、技巧或方法，但是确实有可以掌握的技能。不过，全面而恰当地应用这些技能仍是一门艺术。

在第一部分中，我们从基本概念、原理和发展趋势讲起，这些内容构成系统分析和设计方法的环境，换句话说，基础知识！如果你理解了这些基础知识，就能更好地、更有信心地应用第二部分到第四部分学到的实际工具和技术，你也将能适应新的情况和新的方法。

这一部分包括 3 章。第 1 章（系统分析和设计方法的环境）介绍系统分析和设计的参与者，其中特别强调现代系统分析员作为系统工作的推进器的作用。你也将了解到系统分析员、最终用户、管理人员和其他信息系统专家之间的关系。最后，你将学到如何成为一名系统分析员（如果这是你的目标）。无论你将来做什么，都应理解如何同这些重要的专业人员打交道。

第 2 章（信息系统开发）介绍一个高层的（即通用的）信息系统开发过程。这个过程称为系统开发生命周期。我们将以大多数人实际使用的方式（即系统开发方法学）介绍这个生命周期。这个方法学将成为你学习和应用本书后续章节中讲解的系统分析和设计方法的基础。

第 3 章（项目管理）介绍项目管理技术。所有的系统项目都依赖这些原理。该章介绍项目管理的两种建模技术：甘特图和 PERT 图。这些工具有助于读者安排活动、评估进展和调整计划。

系统分析和设计方法的环境

本章概述和学习目标

本书讲述的是如何将系统分析和设计方法应用到信息系统和计算机应用中。无论你在任何业务中选择什么职业或处于何种位置，你将很可能参与系统分析与设计。你们中的一些人将成为系统分析员（系统分析与设计活动中的关键人员）；而其余人将在项目进入到你们企业或机构时与这些系统分析员一起工作。

本章从4个不同角度介绍信息系统。通过学习本章，你将了解到系统分析和设计方法的环境。

- 定义信息系统以及7类信息系统应用。
- 确定使用或开发信息系统的不同类型的关联人员，并给出每种类型的例子。
- 定义系统分析员在信息系统开发中的独特角色。
- 描述当前影响信息系统开发的业务驱动力。
- 描述当前影响信息系统开发的技术驱动力。
- 简要描述开发信息系统的简单过程。

本章关键术语

信息系统（Information System, IS）是人、数据、过程和信息之间相互作用，收集、处理、存储和提供支持企业运作的信息的集合体。

信息技术（Information Technology, IT）是一个现代词汇，是计算机技术（硬件和软件）和电信技术（数据、图像和语音网络）相结合的产物。

事务处理系统（Transaction Processing System, TPS）是一种捕捉和处理有关企业事务数据的信息系统。

管理信息系统（Management Information System, MIS）是一种提供面向管理的企业业务处理和运作报告的信息系统。

决策支持系统（Decision Support System, DSS）是一种信息系统，辅助制定决策，或者提供制定决策的信息。

主管信息系统（Executive Information System, EIS）是支持主管经理的规划和评估需求的信息系统。

专家系统（expert system）捕捉技术专家的专业知识，然后模拟这些专业知识为非专家服务的信息系统。

通信和协作系统（communication and collaboration system）促进工作人员、合作伙伴、顾客和供应商之间进行更有效的通信，以提高他们协作能力的信息系统。

办公自动化系统（office automation system）支持广泛的企业办公活动，改进工作人员之间工作流的信息系统。

关联人员（stakeholder）是与某个已存在的信息系统或新信息系统有利益关系的人。关联人员可以是技术工作者，也可以是非技术工作者；既包括内部工作人员，也包括外部工作人员。

信息工作者（information worker）指在工作中涉及到创建、收集、处理、分发和使用信息的人。

系统所有者（system owner）是信息系统的发起人和主要倡导者，他们通常在项目的信息系统开发、运行和维护上提供资金。

系统用户（system user）是那些在通常意义上使用信息系统或者受到信息系统影响的“客户”——如收集、验证、录入、响应、存储、交换数据和信息。

知识工作者（knowledge worker）是指其工作基于专业知识的工作者。

远程用户（remote user）是指不在公司办公地点但仍需要访问信息系统的用户。

移动用户（mobile user）是指位置经常变化，但需要从任意地点都能访问到信息系统的用户。

系统设计人员（system designer）是将系统用户的业务需求和约束条件转换成技术方案的技术专家。他们设计满足系统用户需求的计算机数据库、输入、输出、屏幕界面、网络和程序。

系统构造人员（system builder）是根据系统设计人员的设计说明构造信息系统及其构件的技术专家。

系统分析员（system analyst）研究组织存在的问题和需求，确定人员、数据、过程和信息技术如何最大化地为企业做出贡献。

外部服务提供者（External Service Provider, ESP）是指有偿提供他们的专业知识和经验给其他企业，帮助那些企业购买、开发和集成企业信息系统的系统分析员、系统设计人员或者系统构造人员。他们可能属于某咨询机构或服务机构。

项目经理（project manager）是经验丰富的专业人员，主要负责根据进度安排、预算、发布的产品、客户满意度、技术标准和系统质量，计划、监督和控制项目。

电子商务（electronic commerce, e-commerce）是指通过使用因特网购买和销售商品及服务。

电子业务（electronic business, e-business）是指通过使用因特网进行日常的业务活动。

数据（data）是组织内部关于人、地点、事件和事务的重要原始事实。单独的数据并没有什么意义。

信息（information）是为某些人进行处理或重新组织成更有意义的的形式。信息通过数据的组合形成，这种组合期望对接收者有意义。

知识（knowledge）是依据接收者的事实、真理、信仰、判断、经验和专业知识进一步提炼后的数据和信息。理想情况下，信息产生智慧。

业务过程（business process）是响应业务事件（例如订单）的任务。业务过程是完成任务所需要的工作、程序和规则，它独立于自动化或支持它们的信息技术。

持续过程改进（Continuous Process Improvement, CPI）是连续地监控业务过程对降低成本和增加效益方面虽微小但可度量的改善之影响。

全面质量管理（Total Quality Management, TQM）是一种在企业内部促进质量改善和管理的综合方法。

业务过程重构（Business Process Redesign, BPR）是指研究、分析和重新设计企业的基本业务过程，为企业降低成本和/或提高效益。

对象技术（object technology）是一种软件技术，它采用封装了数据和行为的对象来定义系统。对于软件开发人员来说，对象是可复用和可扩展的。

面向对象分析和设计（object-oriented analysis and design）是用于系统开发的一组工具和技术的集合，利用对象技术来构造系统及其软件。

敏捷开发（agile development）是一种系统开发策略，系统开发人员可以从一套相应的工具和技术中灵活地选择最适合完成手边任务的工具和技术。敏捷开发被认为可以在系统开发的产量和质量之间达到最优化的平衡。

企业资源规划（Enterprise Resource Planning, ERP）是一种应用软件，它将信息系统完全集成在一起，提供大部分或者所有核心基本业务功能（包括这些业务功能所需的事务处理和管理信息）。

供应链管理（Supply Chain Management, SCM）是一种应用软件，它通过直接将企业的信息系统与企业的供应商和分销商的信息系统集成，优化从原材料采购到最终产品分销的业务过程。

客户关系管理（Customer Relationship Management, CRM）是一种应用软件，它为客户提供对企业过程的访问，从初始的咨询，直到售后服务和支持。

企业应用集成（Enterprise Application Integration, EAI）是指用来链接应用软件以支持应用软件之间的数据和信息流的过程和技术。EAI 解决方案通常基于中间件。

中间件（middleware）用来在不同应用之间转换和路由数据的（通常是购买的）软件。

系统开发过程（system development process）是一组活动、方法、最佳实践、交付成果和自动化工具的总称。系统开发的关联人员用它们来开发和维护信息系统及软件。

项目管理（project management）是指为了在指定的时间和预算范围内开发出一个可接受的系统而定义、规划、指导、监视和控制项目的活动。

过程管理（process management）是指定义、改进和协调一个组织为所有系统开发项目所选的系统开发方法（开发过程），过程管理关心项目的阶段、活动、交付产品和质量标准一致地应用于所有项目。

系统启动（system initiation）是指用于定义初始业务范围、目标、进度和预算的项目初始规划。

系统分析（system analysis）研究业务问题领域，以推荐改进措施并说明方案的业务需求和优先权的过程。

系统设计（system design）为系统分析阶段确定的业务需求设计（或构造）一个基于计算机技术方案的过程（注意：越来越多的设计采用原型系统的形式）。

系统实现（system implementation）构造、安装、测试和发布一个系统投入生产（即日常运行）的过程。

1.1 产品——信息系统

正如书名所示，本书是一本讲述系统分析和设计方法的书。最终，这是一本讲述为信息系统“分析”业务需求，并“设计”信息系统以满足那些业务需求的书。换句话说，系统分析和设计的产品是信息系统，这个产品在可视化框架结构中表示为图形中央的一个大矩形框。

系统由一组交互的构件组成，它们在一起工作以实现我们所要求的结果。

企业的**信息系统**捕捉和管理数据以产生有用的信息，为企业以及企业的雇员、客户、供应商和合作伙伴提供支持。许多企业把信息系统看成是他们的竞争力或者获取竞争优势的根本要素，绝大多数企业也已经认识到所有的工作人员都需要参与到信息系统的开发中。因此，信息系统开发是一件与每个人都有关系的事情。

信息系统以各种形式和规模出现，它们如此紧密地与它们为之服务的业务系统交织在一起，以至于经常很难区分业务系统与支持业务系统的信息系统。能够将信息系统按照它们提供的功能进行分类就足够了。**事务处理系统**处理企业事务，例如订单、计时卡片、支付和预订。**管理信息系统**使用事务数据产生管理者管理企业所需的信息。**决策支持系统**辅助各种决策人员从可选项中选出决策。**主管信息系统**专门按照主管的特殊信息需求进行裁剪，他们为企业制定规划，并根据规划评估效益。**专家系统**捕捉并加工专家或决策者的知识，然后模拟那些专家的“思想”。**通信和协作系统**提高人们之间的通信和协作水平，无论他们在企业内部，还是在企业外部。最后，**办公自动化系统**辅助雇员生成和共享那些支持日常办公活动的文档。

正如图 1-1 所示，可以从不同的视角审视信息系统，这些视角包括：

- 信息系统的参与者（团队）
- 影响信息系统的业务驱动力
- 信息系统使用的技术驱动力
- 用来开发信息系统的过程

下面我们依次介绍每个视角。

1.2 参与者——系统关联人员

假设你现在的任务是帮助建立一个信息系统，那么系统的**关联人员**是谁呢？信息系统的关联人员大致可以分为 5 类，如图 1-1 左边所示。注意对于同样的信息系统，每类关联人员都有不同的视角。

系统分析员在图 1-1 中是一类独特的关联人员，他们作为协调者，在非技术性的系统所有者和用户与技术性的系统设计人员和构造人员之间起沟通作用。

上述所有的关联人员都有一个共同点——他们都被美国劳工部称为**信息工作者**。信息工作者的生计依赖通过信息做出的决策。如今，超过 60% 的美国劳动力进入信息的生产、分发和使用中。下面更详细地考察 5 类信息工作者。

1.2.1 系统所有者

任何信息系统，无论大小，都必然拥有一个或几个**系统所有者**。系统所有者一般来自管理阶层。对于大中型的信息系统，系统所有者通常是中层或高层经理；对于小型系统，系统所有者可能是中层经理或主管。系统所有者往往对结果感兴趣——系统成本是多少？这个系统将给企业带来多少价值，或者这个系统将给企业带来多大利益？价值和收益可以按不同的方式来衡量，例如：增加企业收益、降低企业费用、系统的费用和利益、增加市场份额、改善客户关系、提高效率、优化决策、更好地遵守法规、减少错误、提高安全性、更大的容量。

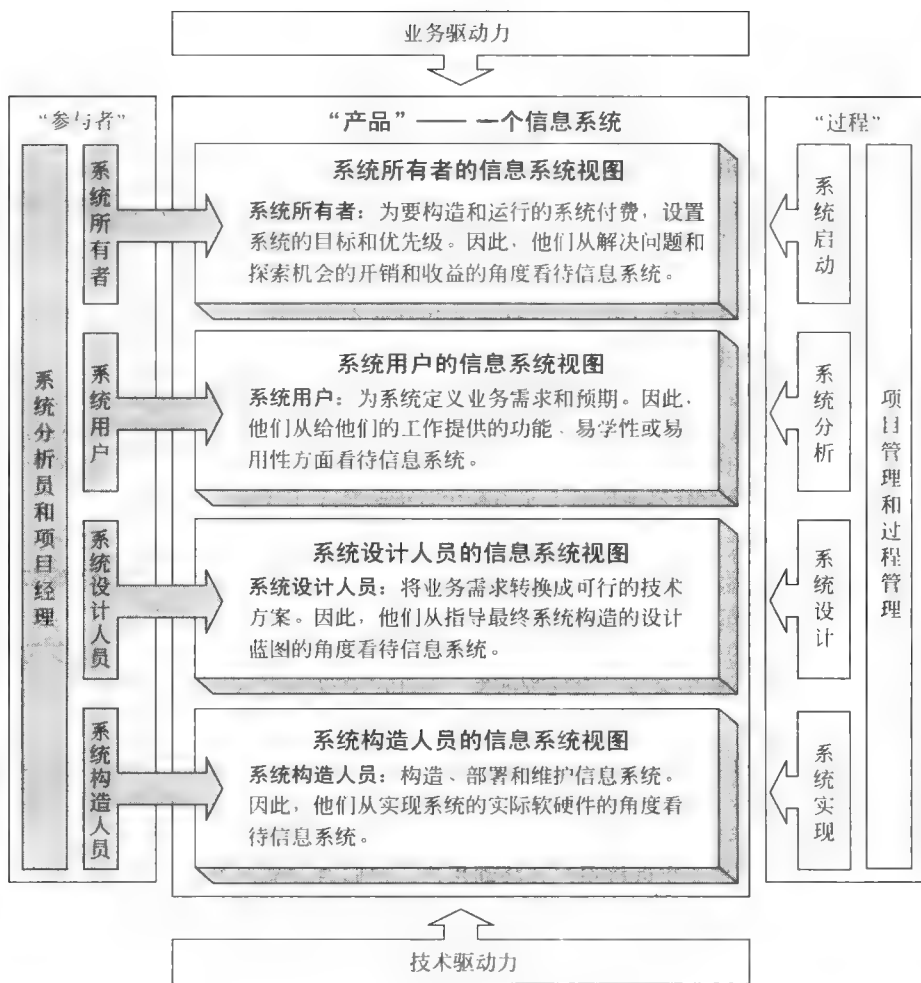


图 1-1 关联人员的信息系统视角

1.2.2 系统用户

任何信息系统中绝大多数的信息工作者均由系统用户构成。同系统所有者不同，系统用户很少关心系统的成本和收益。相反，如图 1-1 所示，他们关心系统提供的功能，系统是否易学易用。尽管这些年来用户变得越来越懂技术，但他们主要关心的仍是如何做好本职工作。因此，同大多数用户的讨论都需要保持在业务需求层面上，而不是技术需求层面上。本书的许多内容都讲述如何有效地沟通信息系统的业务需求。

系统用户分为很多类，每类用户都应直接参与到任何影响他们自身的信息系统开发项目中，下面将简单介绍这些系统用户。

1.2.2.1 内部系统用户

内部系统用户是为之构建信息系统的企业雇员。在大多数企业中，内部用户占据信息系统用户的绝大多数。例如：

- 办事员和服务人员——在一般的企业中，通常由他们处理大部分日常事务，处理订单、发货单、付款之类的事情，录入数据和做文字工作，在仓库履行订单，在商场销售产品。企业中大部分基础数据是由这些工作者搜集或产生的，他们中的许多人除了处理数据以外还要进行体力劳动。面向这些工作者的信息系统往往专注于事务处理的速度和正确性。
- 技术人员和专业人员——主要由业务专家和行业专家构成，他们进行高技术的和专业化的工作。

例如律师、会计师、工程师、科学家、市场分析师、广告设计人员和统计员。他们的工作以公认的知识为基础，因此，有时称他们为**知识工作者**。面向这些信息工作者的信息系统更注重数据分析以及为解决问题产生及时的信息。

- **主管、中层经理和高层经理**——他们都是决策制定者。主管往往关心日常的管理问题；中层经理更关心战术或短期的管理工作和决策；高层经理关心战略或长期计划和决策。为管理人员提供的信息系统往往注重信息获取能力。为了解决问题和制定决策，管理者需要在恰当的时候获得恰当的信息。

1.2.2.2 外部系统用户

因特网拓宽了传统信息系统的边界，将其他企业和直接客户都作为系统用户囊括进来。在现代信息系统中，这些外部用户占据越来越大的比例。例如：

- **顾客**——是指任何购买我们的产品和服务的企业或个人。如今，当顾客直接下订单或者执行销售业务时，他们就成为了我们的信息系统的直接用户，而以往这些工作都需要内部用户的干预。例如，如果你通过因特网购买了某个公司的产品，那么你就成为了那个企业的销售信息系统的外部用户（这里不需要某个独立的企业内部用户来输入你的订单）。
- **供应商**——我们的公司向其购买补给品和原材料的任何单位都称为供应商。如今，这些供应商可以直接同公司的信息系统进行交互，确定我们的供货需求，然后自动生成订单满足需求。不再需要某个内部用户向供应商发出订单。
- **合作伙伴**——我们的公司从其购买服务或者与之合作的公司称为合作伙伴。许多现代企业外包一些基本服务，例如物业管理、网络管理等。企业已经学会了与其他企业合作，事半功倍地生产更好的产品。
- **雇员**——是指那些在外出差或者在家工作的雇员。例如，销售代表通常大部分时间出差在外，许多公司也允许工作人员远程办公（即“在家办公”），以减少费用，提高产量。作为移动用户或者远程用户，这些雇员也同内部用户一样，需要访问信息系统。

外部系统用户越来越多地是指**远程用户**和**移动用户**。他们通过笔记本电脑、掌上电脑和智能电话（有线或者无线方式）连接到信息系统中。为这些设备设计信息系统是我们将在本书中涉及的最新的技术挑战之一。

1.2.3 系统设计人员

系统设计人员是信息系统的技术专家。再次参考图 1-1，系统设计人员对信息技术的选择和使用所选技术设计系统感兴趣。如今的系统设计人员往往专注于某些技术专业。某些读者可能自学过其中的某项专业技术，例如：

- **数据库管理员**——数据库技术专家，负责设计公司数据库以及协调对公司数据库的修改。
- **网络架构师**——网络技术和电信技术领域的专家，他们设计、安装、配置、优化和维护局域网和广域网络，包括到因特网以及其他外部网络的连接。
- **Web 架构师**——为企业设计复杂 Web 站点的专家，包括因特网上的公共 Web 站点，企业内部的内部 Web 站点（称为内联网），以及私有的企业对企业 Web 站点（称为外联网）。
- **图形艺术家**——在如今的 IT 工作者中相对属于新成员，他们是图形技术专家，并擅长设计和构造有吸引力的、易用的系统界面，包括 PC 界面、Web 界面、手持设备界面和智能电话的界面。
- **安全专家**——确保数据和网络安全（以及隐私）的技术和方法的专家。
- **技术专家**——在系统中将使用特定技术应用的专家（例如：某种商业软件包或者某种特殊的硬件）。

1.2.4 系统构造人员

系统构造人员（见图 1-1）代表了另一类信息系统技术专家角色，他们按照系统设计人员的设计说明构造系统。在小型组织，或者对于小型信息系统，系统设计人员和系统构造人员经常是同一组人员。但在大型组织或者大型信息系统中，通常由不同人员来担任。某些读者可能自学过系统构造人员的某项专业技术，例如：

- 应用程序员——他们擅长将业务需求、问题和流程陈述转换成计算机语言。他们开发并测试用于捕捉和存储数据的程序，并为计算机应用程序定位和检索数据。
- 系统程序员——开发、测试和实现操作系统级软件、工具和服务的专家。他们也越来越经常地开发供应用程序员使用的可复用软件“构件”。
- 数据库程序员——数据库语言和技术方面的专家，他们构造、修改和测试数据库结构，以及使用和维护数据库结构的程序。
- 网络管理员——设计、安装、维修和优化计算机网络的专家。
- 安全管理员——设计、实现、维修和从事网络安全和隐私控制的专家。
- Web 站点管理员——编码和维护 Web 服务器的专家。
- 软件集成员——集成软件包、硬件、网络和其他软件包的专家。

虽然本书并不打算直接用于培训系统构造人员，但仍介绍系统设计人员如何更好地与系统构造人员交流。

1.2.5 系统分析员

如上所述，系统所有者、用户、设计人员和构造人员对要构造和使用的信息系统经常存在不同的视点，有些人对普遍性的东西感兴趣，另一些人更关心细节。有些是非技术性人员，而有些人员的技术性很强。那些需要基于计算机的业务解决方案的人和那些懂得信息技术的人之间总存在交流障碍，系统分析员就要沟通这个障碍。你可能（并且可能将）要扮演一个系统分析员的角色，或者与系统分析员一起工作。

如图 1-1 所示，系统分析员需要同系统中的其他所有关联人员交互工作。对于系统所有者和用户来说，分析员确定并验证他们的业务问题和需求；对于系统设计人员和构造人员来说，分析员确保技术方案实现了业务需求，并将技术方案集成到业务中。换句话说，系统分析员通过与其他关联人员的交互推动信息系统的开发。

系统分析员也有几个正式的但常被混淆的工作职称。程序员/分析员（或分析员/程序员）既有计算机程序员的职责，也有系统分析员的责任。业务分析员专注于系统分析和设计中的非技术方面。“系统分析员”的其他同义词包括系统顾问、业务分析员、系统架构师、系统工程师、信息工程师、信息分析员和系统集成师。

有些读者将成为系统分析员，另一些读者将每天同系统分析员一起工作，这些系统分析员通过创建和改进对工作所需数据和信息的访问帮助他们解决商业和工业问题。

1.2.6 外部服务提供者

有一定计算机使用经验的读者可能会奇怪咨询顾问在我们的关联人员概念中处于什么位置。他们的位置在可视化框架中并不那么明显，但他们确实属于其中！任何一个关联人员的角色都可以由内部工作人员或外部工作人员充当。咨询顾问是外部服务提供者（ESP）的一个例子。绝大多数 ESP 是签约的系统分析员、系统设计人员或者系统构造人员，他们为特定的项目提供特殊的专业知识和经验。ESP 包括：技术工程师、销售工程师、系统顾问、签约程序员和系统集成人员等。

1.2.7 项目经理

我们已经介绍了现代信息系统开发中主要的参与者——系统所有者、系统用户、系统设计人员、系统构造人员和系统分析员。最后需要强调一个现实，也就是所有这些人必须作为一个团队一起工作才能构造出对企业有益的信息系统。有团队就需要领导。因此，这些关联人员中的一个或者几个人将承担起项目经理的角色，他们确保系统及时、按预算并以可以接受的质量开发出来。如图 1-1 所示，大多数项目理由有经验的系统分析员担任。但有些组织，项目经理是从我们称为“系统所有者”的人中挑选的。无论如何，大多数组织都已经认识到项目管理是一个专业的角色，它需要专门的技能和经验。

1.3 现代信息系统的业务驱动力

考虑信息系统产品的另一种方式是从企业驱动力的角度来分析。图 1-2 简单地描述了影响信息系统

的最重要的企业发展趋势。许多潮流转瞬即逝，但这里介绍的都是我们认为在未来几年里会影响到系统开发的发展趋势。许多发展趋势是相互关联的，从而形成了一种新的业务哲学，这将影响到接下来许多年每个人的工作方式。

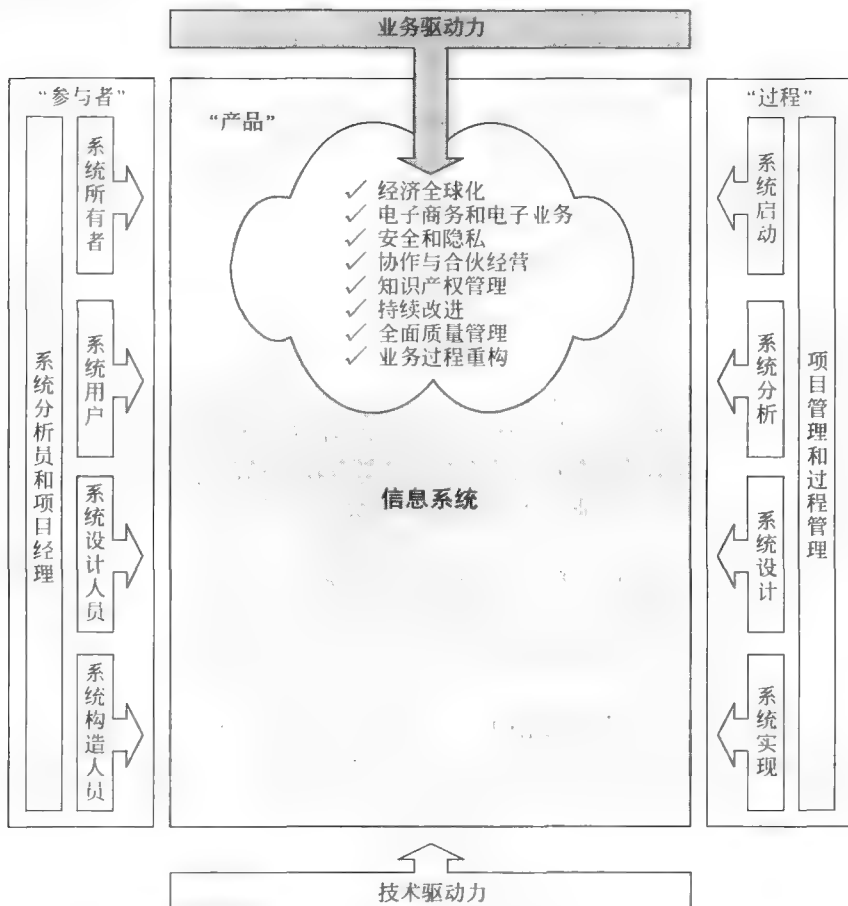


图 1-2 信息系统的业务驱动力

1.3.1 经济全球化

20 世纪 90 年代以来，经济全球化的趋势越发明显。随着越来越多的工业化国家提供低价的或者高质量的产品，竞争演变成全球化。美国企业突然发现市场上出现了许多新的国际竞争者。另一方面，许多美国企业也为它们的产品和服务开拓了许多新的更大的国际市场。大多数企业被迫重组，以便能够全球经济中运营。

经济全球化将如何影响到信息系统中的参与者？首先，信息系统和计算机应用必须国际化，它们必须支持多种语言、货币汇率、国际贸易规则、不同的商业文化的业务方式等。其次，大多数信息系统最终也需要实现信息融合，以实现性能分析和决策支持。语言障碍、货币汇率、跨国的信息规则之类的问题使得融合复杂化。最后，参与者需要同时使用不同语言、方言或俚语的管理人员和用户交流（口头或者书面）。系统分析员在国际上雇用的机会应该会不断地增加。

1.3.2 电子商务和电子业务

由于经济全球化和无处不在的因特网，企业正在改变或者扩展它们的业务模式，以实现电子商务（e-commerce）和电子业务（e-business）。因特网正在从根本上改变企业做生意的规则。我们生活这样一个世界，顾客和企业将越来越期望使用因特网进行商务活动（处理业务事务）。因为商业社会的人

们已经如此熟悉“在网上冲浪”，所以越来越多的企业将 Web 界面作为一种合适的处理企业内部日常事务的体系结构。

有三类基本的电子商务和电子业务信息系统应用：

- 企业形象、产品和服务的营销宣传是最简单的电子商务应用形式，其中万维网仅仅用来向客户“通告”有关产品、服务和公司策略的信息。大多数企业已经实现了这个层次的电子商务。
- 企业对客户（B2C）的电子商务试图为传统的产品和服务提供新的基于 Web 的销售渠道。客户一般可以直接通过因特网查询商品、订购商品并为商品付款。B2C 电子商务的例子有 Amazon.com（销售图书和音像）和 E-trade.com（销售股票和债券）。这两个公司都是建立在 Web 上的企业的例子，它们的竞争者是那些增加了基于 Web 的电子商务前端的传统企业（例如 Barnes & Noble 书店和美林证券）。图 1-3 展示了一个典型的 B2C Web 店铺。



图 1-3 一个电子商务店铺

- 企业对企业（B2B）的电子商务则代表着真正的未来！这是一种最复杂的电子商务形式，并且最终会进化到电子业务——企业之间和企业内部的几乎所有业务完全无纸化和数字化处理。

B2B 电子商务的一个例子是电子采购。企业每年用于购买原材料、设备和补给品的花费常常在上千万或上亿美元。B2B 采购让雇员浏览电子店铺和商品目录，发出购买请求和订单，转发购买请求和订单以获得购买许可，订购商品和服务并为发货的商品和实现的服务付费——所有这些都可以避免传统的耗时费钱的文书工作和官僚作风。主要由于向这些电子商务和电子业务应用发展的趋势，大多数新的信息系统都面向因特网体系结构设计。图 1-4 是一个基于 Web 的采购店铺的例子。

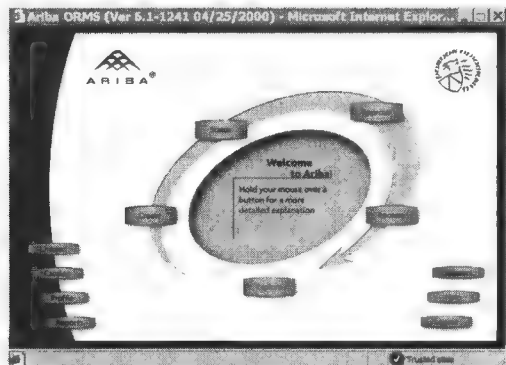


图 1-4 一个电子商务采购店铺

1.3.3 安全和隐私

随着数字经济的发展，公民和组织对如今经济生活中涉及的安全和隐私问题的认识水平都有所提高。安全问题趋于围绕业务的连续性来考虑，也就是说，“当遭到破坏或者灾难（引起业务活动中止的任何事件）时业务如何进行”。另外，企业必须自问，“企业如何保护它的数字财产不受外部侵害？”确实这些问题最终都需要技术来解决，但是，其利害关系则是一个基本的商务问题。

同安全相关的另一个话题是隐私。在数字经济中，顾客对隐私的要求越来越高。政府正在对隐私问题进行立法。随着数字经济的发展，法规可能会越发严厉。当你访问你喜欢的企业站点时，就会发现几乎每个企业都有自己的隐私策略。消费者组织已经开始分析并监视这类隐私策略的执行，保证公司可被审查，并游说政府制定更严格的法律法规。

1.3.4 协作与合伙经营

协作与合伙经营是影响信息系统应用的重要业务趋势。在组织内部，管理层强调打破独立组织部门和职能部门之间的壁垒。管理层谈及“交叉功能”团队，这种团队从多学科的视角相互协作实现共同的业务目标。例如，以前新产品设计主要是工程师的事，如今新产品设计通常都涉及来自许多组织部门的代表构成的交叉功能团队，这些部门可能是：工程部、市场部、销售部、生产部、库存控制部门、分销商以及信息系统部门。

协作的趋势同样扩展到组织以外，包括了其他的组织，有时甚至包括竞争对手。有经济头脑的企业选择在商务活动中以合作伙伴的形式直接协作。Microsoft 和 Oracle 销售相互竞争的数据库管理系统，但 Microsoft 和 Oracle 也相互合作确保 Oracle 应用软件能够在 Microsoft 数据库上运行。两个公司都从这个合作中获得经济利益。按照类似的思路，企业已经认识到信息系统之间互操作将大有益处。

1.3.5 知识产权管理

何谓知识？知识是我们将原始数据处理成有用信息的连续过程的产物。信息系统通过捕捉业务数据（关于产品、雇员、客户等的的数据）收集原始数据，并处理业务事务。数据经过组合、过滤、组织和分析，形成有助于管理人员规划和运行企业的信息。最终，人们将信息提炼成知识。越来越多的企业开始自问，“公司如何管理和共享知识，以获得竞争优势？随着工作人员的变动，企业如何保存它们的知识和经验？”

知识产权管理的需求在很多方面影响着信息系统。尽管我们已经在信息系统中捕捉了（并且仍在继续捕捉）大量的数据和信息，但这些数据和信息在大多数企业中是松散地集成在一起的——数据和信息的冗余和矛盾在信息系统中司空见惯。随着新信息系统的建设，我们越发需要关注数据和信息的集成，这样才能为企业创造和保存知识。这将使系统分析和设计更加复杂。在本书中，我们将介绍许多有助于改进知识管理水平的系统集成工具和技术。

1.3.6 持续改进和全面质量管理

信息系统使业务过程自动化，并支持业务过程的实现。持续过程改进（CPI）仔细检查业务过程，实现一系列的小改进，以便持续改进业务过程。持续改进有助于减少开销、提高效率和增加价值。系统分析员不仅会受到持续过程改进的影响，而且在设计和实现信息系统时要发起或建议这种改进。

另一个业务驱动力是全面质量管理（TQM）。企业已经知道质量是成功的关键，质量管理不仅仅与所销售的产品和服务有关，而且是一种企业中的每个人都对质量负有责任的文化。TQM 的做法要求每个企业职能部门（包括信息服务部门）确定质量指标、度量质量并做出相应的改变以提高质量。

信息系统要满足全面质量管理的要求，所以系统分析员也要满足全面质量管理的要求。我们同大学毕业求职人员进行的讨论表明，对质量管理的“强迫性”态度是成为一位成功的系统分析员（和所有的信息技术专家）的基本素质。在本书中，持续过程改进和全面质量管理将是一个重要主题。

1.3.7 业务过程重构

如前所述，许多信息系统支持或者自动化企业的业务过程。大部分企业正发现这些业务过程几十

年都没有变化，而且那些业务过程非常低效或成本高昂。许多过程过于官僚主义，不能真正地为企业贡献价值。信息系统只是将这些低效率的过程自动化。真正的解决办法就是业务过程重构！

业务过程重构（BPR）涉及对更大系统间的业务过程的根本性改变。实际上，BPR 寻求实现比 CPI 更根本性的改变和改进。在 BPR 过程中，每个过程的各个方面都根据时限、瓶颈、开销和是否真正给组织带来价值（或只是增加了官僚作风）进行分析。业务过程最终按照效率最大化和开销最小化进行重构。

1.4 信息系统的技术推动力

信息技术的进步也是信息系统的推动力（如图 1-5 所示）。一方面，过时的技术会带来很大的问题，从而驱动信息系统项目开发。另一方面，新的技术引发新的机会。下面我们就来介绍几个影响信息系统的技术因素。

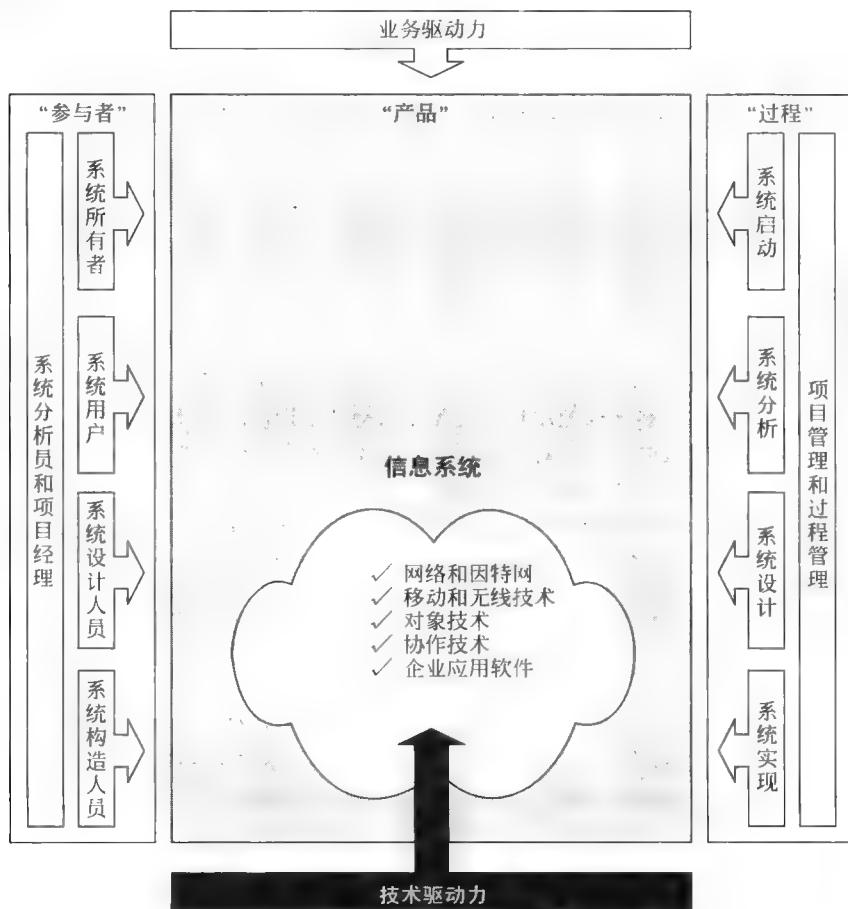


图 1-5 信息系统的技术驱动力

1.4.1 网络和因特网

Sun 计算机公司具有超凡魅力的 CEO，Scott McNealy 有一句名言：“网络就是计算机。”人们都知道如今的信息系统是构建在网络体系结构上的，由局域网和广域网构成。这些网络包含了大型主机、网络服务器、各种台式机、笔记本电脑和掌上电脑的计算机客户端。但如今，最普遍的网络技术是基于因特网的技术。下面列出了一些读者应该了解（即使没有这方面的技能）的与因特网有关的技术。

- xHTML 和 XML 是编写 Web 页面和因特网应用程序的基本语言。扩展超文本标记语言（xHTML）是 HTML 的第二版，用于构造 Web 页面。扩展标记语言（XML）是在因特网上有效传输数据内

容及其语义的语言。介绍 xHTML 和 XML 的课程已经成为绝大多数信息系统和信息技术大学课程表中的核心内容。

- 脚本语言是专门为因特网应用程序设计的简单编程语言，例如：Perl、VBScript 和 JavaScript。这些语言正逐步被大学的 Web 开发和程序设计课程所讲授。
- Web 专用语言（例如 Java 和 Cold Fusion）专门用来构造复杂的基于 Web 的应用程序，这些程序往往涉及多个服务器和 Web 浏览器。这些语言在大学编程课表中也是必备的。
- 内联网是供组织内部雇员使用的私有因特网。内联网具有因特网的界面，但是安全功能和防火墙的限制只能由雇员使用。
- 外联网同内联网一样，也是私有的因特网，但外联网用于特定的组织之间。只有那些通过合法认证的企业雇员才能访问和使用外联网。例如，汽车厂商（如雪佛莱）可能会为它的经销商建立一个外联网，通过这个外联网，厂商可以交流零件、问题、促销等信息。
- （公司）门户是一个“主页”，它可以根据使用者的不同需求进行定制。例如，门户技术可以定义一些 Web 页面，它们为同一公司中的不同角色人员提供恰当的信息和应用软件。每个人的角色决定了他可以从其 Web 页面使用的信息和应用软件。这里角色可以是“顾客”、“供应商”，以及不同类型的“雇员”。门户也可以将公共因特网、私有的内联网和外联网集成到每个人的用户主页中。
- Web 服务是最新的技术潮流。它是指可复用的、基于 Web 的程序，这些程序可以从任何其他因特网程序中调用。例如，假设你需要编写一个程序通过 Web 接受信用卡支付。当然，你可以自己编写、调试和测试信用卡验证程序。但另一种方法是购买使用某个 Web 上的信用卡验证程序的使用权。这样你就不必维护信用卡认证代码，只需要从你的程序中“调用”Web 服务就可以了，就像调用一个内部子程序一样。当然，你将为使用 Web 服务而付费，因为需要有人编写这些 Web 服务程序。



无线掌上电脑

以上仅仅是你应当学到的网络和因特网技术中很小的一部分。但你必须认识到因特网的快速发展特性，并接受在不久的将来这些技术和其他技术将不断出现和消失这一现实。

1.4.2 移动和无线技术

移动和无线技术必将极大地改变下一代信息系统。掌上电脑或者个人数据助理（PDA）在信息工作者中很常见。这些设备正逐渐具备了无线功能，提供 Web 访问和电子邮件。移动电话也越来越多地增加了因特网和电子邮件功能。如今，出现了一些集成的设备（如智能电话），它们将 PDA 的功能和移动电话的功能集成到单个设备中，如右图所示。对于那些喜欢独立设备的人，蓝牙之类的技术使得独立的设备可以作为一个逻辑设备互操作，同时仍保持每个设备的特点和优点。

另外，越来越多的笔记本电脑配备了无线和移动功能，使得信息工作者携带电脑更容易，并保持同信息系统的连接。所有这些技术趋势将深远地影响新信息系统的分析和设计。渐渐地，无限访问能力必须作为前提条件，所以移动设备和屏幕尺寸的限制必须在信息系统中有所考虑。本书将讲授并演示用于移动应用软件设计的工具和技术。

1.4.3 对象技术

绝大多数现代的信息系统都是使用对象技术构造的。如今最常用的编程语言也是面向对象的，例如 C++、Java、Smalltalk 和 Visual Basic .NET。使用对象技术，程序员可以利用称为对象的软件部件构造软件。（我们将在本书后面更详细地介绍对象。）面向对象软件相比非面向对象软件有两个基本优点。第一，对象是可复用的。一旦对象被设计和构造出来，它们就可以被复



智能电话

用于多个信息系统和应用软件中。这减少了开发未来的软件的时间。第二，对象是可扩展的。它们可以很容易地被修改和扩展，而不会影响任何以前使用这些对象的应用软件。这减少了维护和改进软件的生命期的费用。

对象技术对系统分析和设计影响重大。因此，面向对象分析和设计方法已经成为构造绝大多数现代信息系统的首选方法。所以，我们将在本书中融入面向对象分析和设计工具与技术，以使读者在未来的职场中获得一定的竞争优势。同时，结构化工具和方法仍很重要。两种方法我们都提倡，我们将介绍何时以及如何组合结构化和面向对象工具和技术用于系统分析和设计。当写作本章时，有一种称为敏捷开发的技术在资深分析员中很受青睐，这些分析员已经厌倦了那些坚持只使用某一种方法工具和方法的过于约定俗成的方法。简单地说，敏捷方法可以看成是一个包括不同工具和技术（结构化的、面向对象的等）的工具箱，可以从中选择最合适的工具或技术来解决在系统分析中遇到的任何问题或需要。

1.4.4 协作技术

另一个重要技术趋势是协作技术的使用。协作技术是指那些提高人际交互和团队工作能力的技术，电子邮件、即时消息、群件和工作流系统是4类重要的协作技术。

每个人都知道电子邮件是什么，但电子邮件在信息系统开发中的重要性正在变化。现代信息系统越来越是电子邮件使能的，也就是，电子邮件功能是应用软件的基本功能，使用电子邮件并不需要切换到某个专用的电子邮件程序（例如 Outlook）。应用程序只是调用用户的（或者组织的）默认电子邮件程序，发送或者接收相关消息。

与电子邮件相对应的技术是即时消息（例如，AOL 的 Instant Messenger 和微软的 MSN Messenger Service）。即时消息在因特网上的公共（和私有）“聊天室”中很流行，但即时消息也慢慢地被集成到企业信息系统应用中。例如，即时消息可以为某个企业应用程序的帮助系统实现立即响应功能。想象一下当使用企业应用软件时，能够及时地发送消息给帮助中心并及时接收到响应，工作效率和服务水平的提高将十分明显。

最后，群件技术使得人们可以在项目和任务中协作，而无论他们的地理位置在哪里。群件技术的例子有：Lotus 的 SameTime 和微软的 NetMeeting。使用这类群件软件后，多个人可以进行网络会议并共享软件工具。同电子邮件和即时消息一样，群件功能也可以在相应的企业应用软件中内建。

显然，系统分析员和系统设计人员需要把这些革命性的协作技术构建到他们的应用软件中。

1.4.5 企业应用软件

几乎所有的组织（无论规模如何）都需要一套核心企业应用软件来保证业务运行。如图 1-6 所示，对于大多数企业来说，核心应用软件包括财务管理、人力资源管理、市场和销售以及运行管理（库存或生产控制）。曾经绝大多数企业自己构造大部分或者全部核心企业应用软件。但如今，企业常常是购买、安装和配置核心企业应用软件，并把它们集成到组织的业务过程中。为什么？因为在不同的组织和行业中，核心企业应用软件的相似性大于差异。

如今，这些“内部的”核心应用与其他企业应用软件互补，那些软件将企业的业务过程同它的供应商和客户的业务过程集成到一起，称为客户关系管理和供应链管理（见图 1-6）。

这种使用购买的企业应用软件的趋势对系统分析和设计的影响很大。对于任何组织来说，购买和安装的企业应用软件从来都不能完全满足信息系统的所有需求。因此，就要求系统分析员和其他开发人员开发增值应用软件以满足企业的其他需求。但购买和安装的企业应用软件成为了一项技术限制条件，用户的任何软件都需要正确地同它们集成和交互。这通常称为系统集成，这也正是大多数读者毕业后将投入的企业和系统环境。下面我们就简要地介绍某些较常见的企业应用软件，以及它们对系统分析和设计的影响。

1.4.5.1 企业资源规划

如前所述，大多数企业的核心业务信息系统在内部增量式地开发了许多年。每个系统都有自己的文件和数据库，并且松散地集成在一起。20 世纪 90 年代，企业试图集成它们的遗留信息系统，但效果

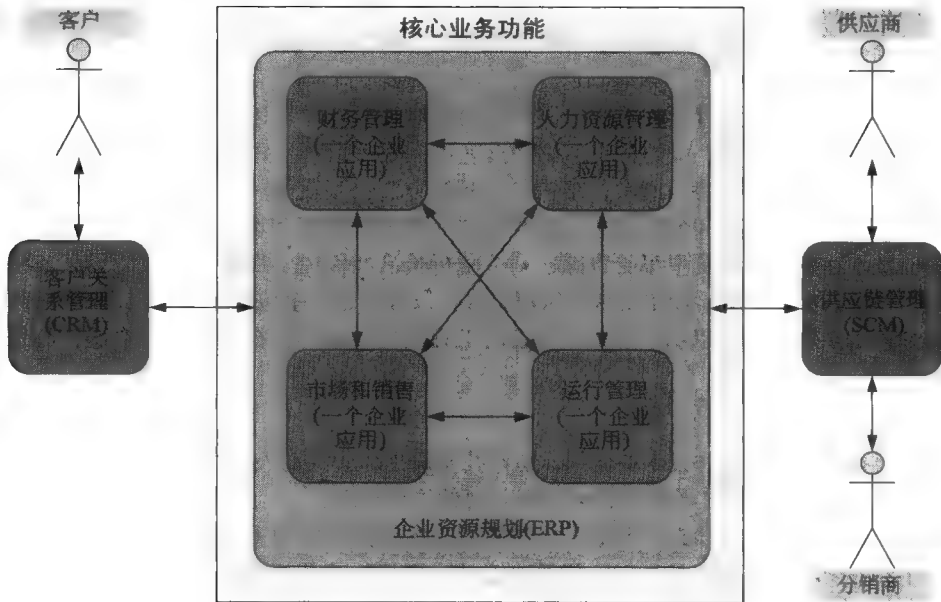


图 1-6 企业应用软件

通常不好。理想情况下，组织应当从头重新开发它的核心业务应用系统，使之成为一个单一的集成信息系统。不过，很少有企业有足够的资源尝试这种做法。由于认识到大多数企业需要的基本应用系统是类似的，软件行业对这个问题开发了一个方案——企业资源规划（ERP）。ERP 产品围绕一个由基本业务功能共享的公共数据库建造。下面列出了一些 ERP 软件供应商：SSA、Oracle/PeopleSoft、SAP AG（市场领导者）。

ERP 方案为企业提供了核心信息系统功能。为了同 ERP 方案集成，企业必须重构它的业务过程。大多数企业还需要使用定制软件来补充 ERP 方案，以满足企业或行业特殊的业务需求。对大多数公司来说，一个 ERP 实现和集成通常是该公司承担过的规模最大的信息系统项目。它可能需要花费几千万美元，需要一支由管理人员、用户、分析员、技术专家、程序员和咨询顾问构成的团队。

ERP 应用对系统分析员之所以重要有几个原因。第一，系统分析员可能要决策选择和购买 ERP 软件。第二，也是更常见的情况，系统分析员经常要定制 ERP 软件，以及重新设计业务过程使用 ERP 软件。第三，如果在使用 ERP 核心软件的组织中开发定制应用软件，ERP 系统的体系结构将极大地影响定制应用软件的分析和设计，因为定制应用软件必须同 ERP 系统互操作。

1.4.5.2 供应链管理

如今，许多组织都在努力开发那些扩展支持核心业务功能以外功能的企业应用。公司扩展它们的核心企业应用软件，以便与公司的供应商和分销商的系统实现互操作，更有效地管理原材料和产品在相关组织之间的流动。这些供应链管理软件利用因特网作为集成和通信的手段。

例如，图 1-7 展示了一个逻辑供应链，该供应链结束于属于某个代销商的餐馆。注意，这个供应链包含了多个企业和运输者，目的是实现其最终目标：确保餐馆有足够的食品供应。在这个供应链中任何一段出现延迟或问题都将影响其他部分或者影响整体。因此，这些企业将使用 SCM 软件技术实现供应链管理，以便规划、实现和管理这个供应链。以下列出了一些供应链管理软件供应商：i2 Technologies、Manugistics、SAP、SCT。（注意，有些 ERP 软件商正扩展它们的 ERP 软件，使其包含 SCM 功能。）

SCM 应用对系统分析员很重要，其原因同 ERP 应用的情况一样。作为一位分析员，你可能要评估和选择 SCM 软件，或者实现（定制）这些软件以满足企业需求。另外，你可能参与重构现有业务过程，以便能够恰当地适应 SCM 软件。

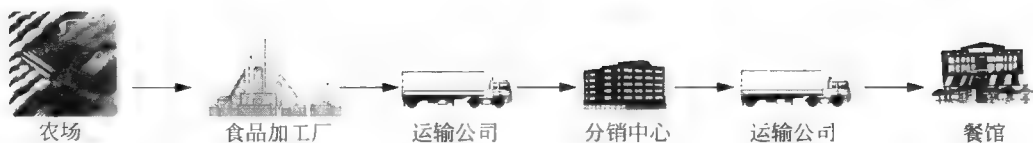


图 1-7 供应链

1.4.5.3 客户关系管理

许多公司发现高度聚焦的客户关系管理可以提高客户忠诚度，从而增加销量。因此，许多企业正在实现客户关系管理（CRM）方案，使客户可以通过因特网实现自我服务。所有 CRM 方案的主旋律都是对“客户”的关注。CRM 不仅关注提供有效的客户查询响应和帮助，而且辅助企业更好地建立客户信息数据库，以便改进客户关系和市场工作。以下列出了一些有代表性的 CRM 软件供应商：BroadVision、E. piphany、Kana、Amdocs、Oracle/PeopleSoft、Siebel（市场领导者）、SAP。同 SCM 技术的情况一样，许多 ERP 供应商正在开发或者获取 CRM 功能来补充（和扩展）它们的 ERP 方案，而且更多的 CRM 软件商将被淘汰（通过兼并或者倒闭）。

CRM 技术对系统分析员的影响与 ERP 和 SCM 技术的影响完全一样。在许多企业中，新的应用系统必须提供同某个核心的 CRM 企业应用软件接口。

1.4.5.4 企业应用集成

许多公司面临的重要挑战是集成它们现有的遗留系统和新的应用软件（例如 ERP、SCM 和 CRM）。任何想通过因特网做销售的公司也将必须面临把它的信息系统与其他公司的信息系统集成的挑战。为了应对这个挑战，许多企业寻求企业应用集成软件来解决。企业应用集成（EAI）要将（购买和内部开发的）应用软件链接起来，以便能够透明地互操作，其概念如图 1-8 所示。下面列出一些提供 EAI 工具的供应商：BEA Systems、IBM（MQSeries）、Mercator Software、TIBCO Software。

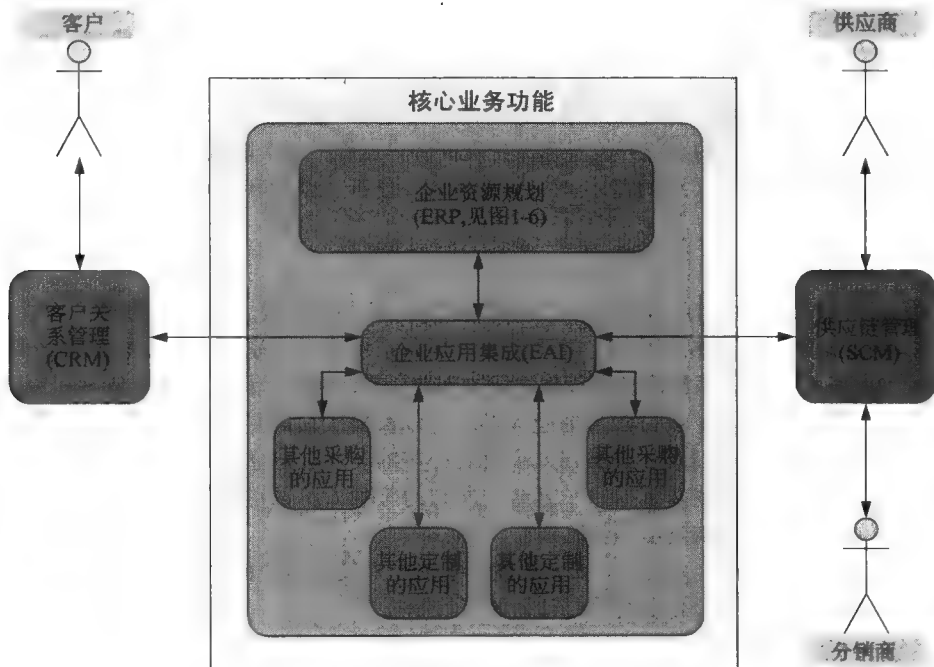


图 1-8 企业应用集成

如今，当开发一个新的信息系统时，它都必须同以前所有的信息系统集成。这些“遗留的”信息系统可能是购买的，也可能是内部构造的。无论如何，系统分析员和其他开发人员都需要为新开发的

信息系统考虑应用集成问题。EAI 技术是集成需求的核心。

1.5 过程——系统开发过程

到目前为止，我们已经了解了不同类型的信息系统，开发这些系统时涉及的参与者，影响信息系统开发的几个业务和技术驱动力。本节将学习另一个信息系统视角——开发信息系统的“过程”。

大多数组织都有一个正式的系统开发过程，其中包括一套标准的过程和步骤，这些过程和步骤在任何系统开发项目中都需要遵循。尽管这些过程对不同组织差别很大，但仍能发现共同特点：大多数组织的系统开发过程都遵循一种问题解决方法。该方法一般包括以下通用问题的解决步骤（见表 1-1）。

表 1-1 通用问题解决步骤和过程之间的关系

简化的系统开发过程	通用问题解决步骤
系统启动	1. 确定问题（也包括规划问题方案）
系统分析	2. 分析和理解问题 3. 确定方案需求和预期
系统设计	4. 确定替代方案，选择“最佳”方案 5. 设计所选方案
系统实现	6. 实现所选方案 7. 评估结果（如果问题没有得到解决，回到第 1 步或第 2 步）

图 1-9 增加了一个系统开发过程视角，在本书中，当我们学习开发过程、工具和技术时，都将（以恰当的细化程度）使用它。为简单起见，我们的初始问题解决方法由 4 个阶段构成，任何系统开发项目都必须完成这 4 个阶段，它们分别是：系统启动、系统分析、系统设计和系统实现。表 1-1 列出了以上通用问题解决步骤和过程之间的关系。

注意任何系统开发过程都必须以项目为基础进行管理。前面提到至少有一个关联人员承担项目经理的责任，确保系统及时、按预算并保质地开发出来。管理项目的活动称为项目管理。相应地，在图 1-9 中增加了一个项目的管理的过程。另外，为了保证所有的项目都按照同样的开发过程管理，我们还包括一个过程管理活动。注意项目管理和过程管理覆盖所有的过程阶段。

下面简要地介绍图 1-9 中的系统开发过程，扩展读者对过程中每个阶段和活动的理解。给定一个需要解决的问题，或者一个需要实现的需求，在系统启动、分析、设计和实现阶段我们都将做什么？每个阶段将涉及哪些人？

1.5.1 系统启动

信息系统项目通常都很复杂，需要投入大量的时间、精力和金钱。要解决的问题经常表述含糊，这意味着初始的方案可能不成熟。因此，需要仔细地规划系统项目。系统启动阶段确立项目范围和问题解决计划。如图 1-9 所示，系统启动阶段确立解决问题所需的项目范围、目标、进度和预算，或项目所带来的机会。项目范围定义了项目涉及的业务领域，以及要实现的目标。范围和目标最终影响资源投入，也就是进度和预算，它们对于成功完成项目是必需的。通过建立对应初始范围和目标的项目进度和预算，也就建立了一个基线。依据这个基线，所有的关联人员都可以接受这样的现实，未来对范围和目标的任何修改，都将影响进度和预算。

图 1-9 也显示了项目经理、系统分析员和系统所有者是系统分析中的主要关联人员。本书将向读者讲授启动系统项目以及建立一个合适的项目计划的工具和技术。

1.5.2 系统分析

系统开发过程的下一步是系统分析。系统分析的目的是为项目团队提供对触发项目的问题和需求更全面的理解。因此，需要研究和分析业务领域（项目范围——按照系统启动阶段的定义），以获得对有什么、没有什么以及需要什么等内容更深入的理解。如图 1-9 所示，系统分析阶段要求同系统用户一起工作以便清楚地定义购买或开发的新系统的业务需求和预期。另外，如果进度和预算不足以实现所

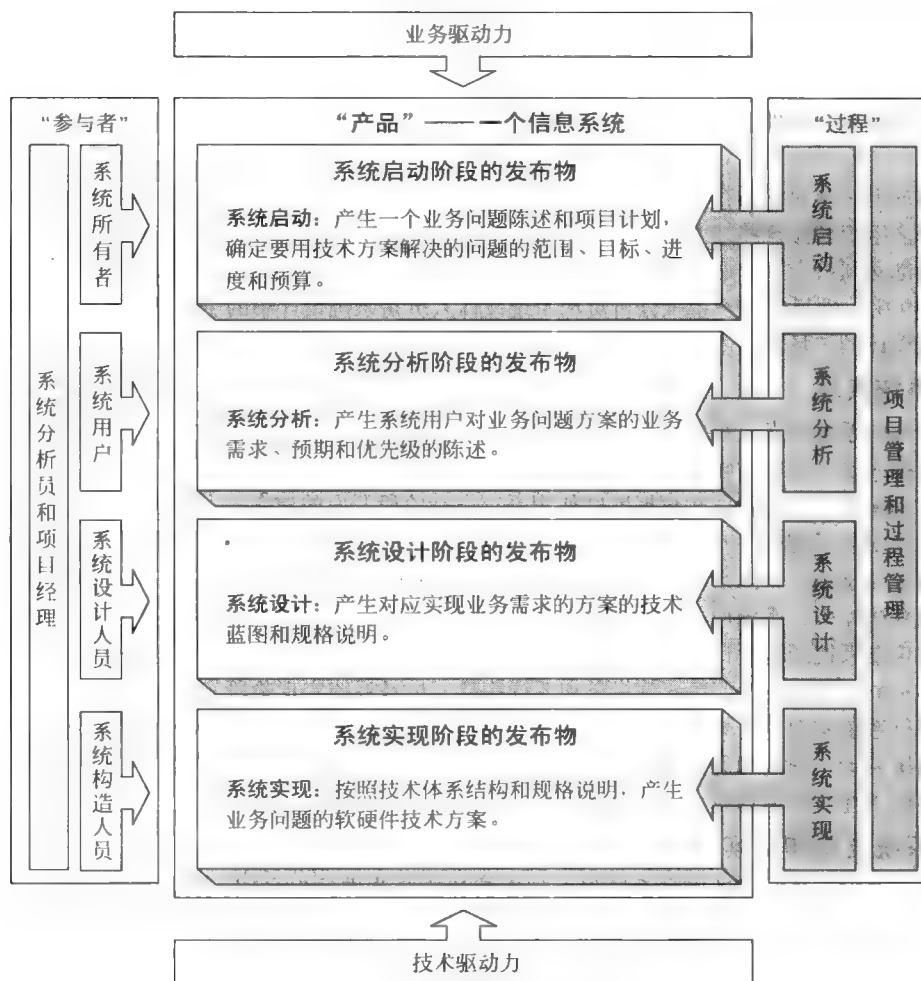


图 1-9 系统开发和问题解决

有的预期，也需要定义业务的优先权。

回顾本章前面讨论的业务驱动力。这些（以及未来的）业务驱动力对系统分析的影响最大，系统分析通常定义对应业务驱动力的业务需求。例如，我们讨论了电子业务和电子商务的发展趋势，这种业务驱动力可能影响到任何信息系统的业务需求，引导我们建立在 Web 上处理所有业务事务的项目目标。

系统分析阶段结束后，通常需要修改以前（系统启动阶段）产生的许多发布产品。分析可能揭示出需要修改业务范围或项目目标，也许现在我们觉得项目的范围太大或者太小。相应地，项目的进度和预算也可能需要修正。最后，项目本身的可行性也会成问题。项目可能要被取消或者继续到下一阶段。

在图 1-9 中，项目经理、系统分析员和系统用户是系统分析中的主要关联人员。一般来说，分析的结果需要给系统所有者总结汇报并进行必要的解释，系统所有者将为设计和实现信息系统以实现业务需求付费。本书将向读者介绍许多用于系统分析和记录用户需求的工具和技术。

1.5.3 系统设计

给定某个信息系统的业务需求，就可以进行系统设计。在系统设计期间，我们最初需要探索不同的技术方案。任何问题至少有一种方案。例如，大多数公司需要在购买一个足够好的方案和构造一个

定制方案之间做出选择。

一旦选定并批准了某个技术方案，系统设计阶段就要开发实现最终方案所需的技术蓝图和规格说明。这些技术蓝图和规格说明将被用来实现信息系统所需的数据库、程序、用户界面和网络。如果选择购买软件而非构造软件，那么蓝图将说明购买的软件如何集成到企业中，以及如何同其他信息系统集成。

回顾本章上一节讨论的技术驱动力。这些（以及未来的）技术驱动力紧密地影响着系统设计过程和决策。许多组织根据这些技术驱动力定义了通用的信息技术体系结构。相应地，所有新的信息系统的系统设计都必须符合这个标准的 IT 体系结构。

在图 1-9 中，项目经理、系统分析员和系统设计人员是系统设计中的主要关联人员。本书将介绍许多进行系统设计的工具和技术。

1.5.4 系统实现

我们简化的系统开发过程的最后一步是**系统实现**。如图 1-9 所示，系统实现阶段构造出新的信息系统，并将其投入使用。新的硬件和系统软件都在该阶段安装和测试，购买的数据库和应用软件也在这个阶段安装和配置，定制软件和数据库则使用系统设计阶段开发的技术蓝图和规格说明进行构造。

当构造或安装系统构件时，必须独立测试它们。同时，必须测试整个系统，以确保其正常工作并满足用户需求和预期。一旦系统经过了全面测试，就可以投入运行。以前系统的数据可能需要转换或输入到启动数据库中，系统用户必须接受培训以便正确使用系统。最后，可能还需要实现某种从旧的业务过程和信息系统的转换计划。

在图 1-9 中，项目经理、系统分析员和系统构造人员是系统实现阶段的主要关联人员。虽然本书将介绍一些进行系统实现的工具和技术，但这些方法主要在程序设计、数据库和网络课程中介绍。本书的重点是系统启动、分析和设计技术，但也会介绍一些大多数系统分析员常用的系统实现工具和技术，这些内容一般不在这类信息技术课程中介绍。

1.5.5 系统支持和持续改进

如果不简要地承认实现的信息系统将面临支持和持续改进的生命期，我们就是失职。但这项内容位于图 1-9 中的什么位置呢？它确实在图中！但有点含糊。

实现后的信息系统很少是完美的。用户将会发现错误（bug），你也经常会发现需要注意和修改的设计和实现缺陷。另外，业务需求和用户需求时常变化。因此，有必要持续地改进信息系统直到它退役为止。那么系统支持和变更在我们的开发过程中处于什么位置呢？

系统支持和改进是另一个项目，有时称为维护或提升项目。这样一个项目应该遵循为其他项目定义同样的问题解决方法。唯一的差别是完成项目所需的努力和预算。许多阶段会很快完成，特别是当原始的关联人员在最初开发时恰当地对系统进行文档记录时。当然，如果他们没有这样做，系统改进项目会花费更多的时间、精力和金钱。我们将在本书中给读者讲授的内容中大部分的目的都是帮助读者正确地记录信息系统，以便明显地减少支持和改进信息系统的生命期费用。

复习题

1. 企业为什么需要信息系统（IS）？
2. 为什么系统分析员需要知道企业中关联人员是谁？
3. 谁是信息系统中的典型关联人员？他们的角色是什么？
4. 请解释如果信息系统缺少系统所有者结果会怎样？
5. 内部用户与外部用户的区别是什么？请举例。
6. 系统分析员的角色与其他关联人员的角色的区别是什么？
7. 除了应该拥有商业和计算知识外，为有效地完成工作，系统分析员还需要哪些基本技能？
8. 如今的信息系统的业务驱动力有哪些？
9. 电子商务和电子业务的差别是什么？
10. 信息和知识的区别是什么？
11. 如今的信息系统最重要的技术驱动是什么？
12. 系统开发过程的四个步骤是什么？每一步做什么？
13. 为什么系统启动阶段在系统开发过程中很重要？

问题和练习

- 假设你是一位系统分析员，正为一家拥有销售机系统的个体的 brick-and-mortar 零售商店做需求分析。确定典型的内部用户和外部用户可能包括哪些人。
- 假设你是一个咨询公司的系统分析员，被委派去帮助一位地区银行的首席执行官（CEO）。银行最近实行一项减少一半员工的计划（包括信贷员）作为保持利润的一个策略。结果，银行发生了严重的信贷请求积压问题，因为信贷员人数太少，不足以完成检查、批准或驳回信贷的工作。银行的 CEO 对能够不增加信贷员而加快审批过程的方案感兴趣，并请你们公司提供建议。你可能会给这个银行推荐哪类系统？
- 通信和协作系统是如何提高工作效率的？正被越来越多的单位使用的通信和协作系统有哪些？
- 单位中的职员一般使用哪类信息系统，为什么？
- 随着信息系统复杂程度的增加，涉及从这些系统中访问和使用数据的道德问题也不断增加。这些道德问题都是什么？
- 什么是企业对客户（B2C）和企业对企业（B2B）的 Web 应用？每类应用列举几个例子。
- 系统开发过程和方法可能会差别很大。假定你是一个系统集成公司的承包人，说出并简要解释本书中描述的系统开发过程的“基本”阶段，以及哪些阶段在任何项目中都需要实现？
- 你的公司同个本地电影公司签约，链接他们的系统，以使它们可以有效地一起工作。他们的系统包括几个现有的遗留系统，系统由不同的开发人员
- 在不同的时候使用不同的语言和平台开发，还包括几个新应用系统。这类链接称为什么？你最可能使用哪类工具？举几个工具的例子。
- 你的公司要求你开发一个新的基于网站的系统来替代现有系统。对现有系统的业务需求和功能的改变很小。你建议使用哪种系统开发过程，为什么？
- 你最近加入一家零售公司，该公司最近购买并兼并了一个商业工业供货库房。你领导一个项目，开发一个统一的库存跟踪系统。你建议使用哪种系统开发过程，为什么？
- 你的公司总裁在一次会前坐在你旁边，告诉你们人们一直说客户需要安装一个 CRM，但他不知道这是什么。然后总裁要你在 30 秒内用非技术的方式解释它。
- 行业研究指出移动和无线技术已经成为设计新的信息系统的主要技术驱动力之一。这是为什么？有什么影响？
- 简要解释 Web 服务对网站开发的影响。举几个关于 Web 服务的例子。
- 说出以下活动属于开发过程的哪个阶段：
 - 开发技术蓝图或设计文档
 - 项目计划
 - 集成测试
 - 同系统用户交流以定义业务需求
- 面向对象软件技术相对于结构化软件技术最主要的两个优点是什么？

项目和研究

- 调研 IT 从业人员的平均（中等）工资，可以使用不同的方法获得这些信息，例如查询网站上发布的 IT 从业人员工资调查结果。也可以查询报纸、商业杂志或网上的分类广告。
 - 系统分析员、设计人员和开发人员的一般工资之间有明显差别吗？
 - 粗略地说，这些不同人员的一般工资之间差别有多大？
 - 你认为造成差别的原因是什么？
 - IT 从业人员工资之间存在性别歧视吗？讨论你发现的任何趋势及其含义。
- 联系几个本地或地区单位的首席信息官（CIO）或高级 IT 经理。询问他们关于在其单位中使用的系统开发过程和方法，以及为什么他们使用那些方法。
 - 描述并比较你发现的不同方法。
 - 你认为哪种方法是最有效的方法？
 - 为什么？
- 职业选择和个人技能：
 - 在你上学期间，如果你必须在系统分析员、系统设计人员和系统构造人员之间做出选择，你选择哪一个？
 - 为什么？
 - 现在，将一张纸分成两列。在一边，分三组分别列出你认为对于系统分析员、系统设计人员和系统构造人员最重要的技能和品德。在另一边，列出至少 5 项你最擅长的技能和品德，然后将它们映射到这三组中的每一项。你拥有的

技能和品德映对哪一组共性最多？

- d. 这一组是你在问题 3a 中选择的那一个吗？你认为为什么是（或不是）？
4. 你学校图书馆中应该有几十年前的杂志和期刊，或者可以登录提供这些资料的联机搜索服务。在信息技术杂志上找几篇关于系统分析的文章，至少 25 年前的几篇文章。
 - a. 比较最近和以前的文章。从这些文章中显示出 25 年前系统分析员所需的典型知识、技能、能力和/或经验与现在所需的相比是否有很大不同？
 - b. 如果你发现某些差异，你认为哪些是最重要的？
 - c. 你认为导致这些差异的原因有哪些？
 - d. 现在掏出你的水晶球，预言 25 年后的情况。

小型案例

1. 你认为今后 10 年技术上有哪些可能的改进？20 年或 30 年后呢？调研一种处于研发阶段的新的有趣的技术。使用电影剪辑或者 PowerPoint 准备一个关于这项技术的幻灯片，并在班上介绍。提交一篇关于这项新技术可能对社会和/或商业的影响的短文。
2. 考察外包：至少部分开发过程经常被外包。事实上，如今项目领导必须能够应对地理上分离的团队以及时限和资源限制。外包提高了效率，给社会带来了经济增长。但是，这些增长并不会很快实现，对社会的负面影响是外包对职业前景有重要影响。Mankiw 博士是 Bush 总统的经济顾问，由于他公开鼓吹外包的好处而受到大量的批评。你认为对于一个企业来说外包是好还是不好？你认为外包的公司会出现道德上的两难吗？查找至少两篇关于外包的影响的文章，然后拿到班上

团队和个人练习

1. 两人一组。第一个人将决定一个他/她希望完成的任务。例如，削铅笔或者写下教授的名字。任务要简单而直接。这个人要在纸上只用图不用语言表达他/她希望做的事，并把图纸交给第二个人。然后第二个人完成这个任务。
2. 从这个练习中你发现了什么？第二个人理解第一个人所要的东西花了多长时间？存在错误交流

你预测现在的系统分析员与 25 年后的系统分析员有什么不同？

5. 搜索网络或者你的图书馆中的商业期刊，例如福布斯杂志，了解 3 ~ 4 个大型公司的首席信息官的信息。
 - a. 你找的 CIO 属于哪个行业领域、公司？
 - b. 对你搜索的每位 CIO，在成为 CIO 之前的主要经历是什么；或者说，他们是否有信息技术背景、商业背景或者二者都有？
 - c. 对每位 CIO，受教育程度如何？
 - d. 他们做 CIO 多少年了，每人大约在几个不同的公司工作过？
 - e. 基于以上搜索结果，要成为 CIO 需要什么知识和技能？为什么？

分享。

3. 你是一个网络管理员，作为你的部分工作，你要监视雇员的电子邮件。你发现为了能够更快地完成项目并满足预算，你的老板正在对你公司开发的一个系统偷工减料。结果系统存在一个错误，如果超过 20 个人同时上网，这个错误可能会导致网络崩溃。而客户预期在任何时候大约有 12 个人上网。你可以确信，很明显你的老板也这样认为，在这个项目被接受之前（如果会被接受的话）顾客是不会发现这个问题的。你将如何做？
4. 系统分析员必须既是技术专家，又能够同客户成功地交流。开发一个新的系统需要全面地理解用户需求。用户经常并不知道（技术上）可以提供什么，甚至他们想从系统得到什么。好的交流的特点是什么？

吗？记录下你的思考和观察，并同班上同学分享。

3. 个人练习：想象一个十分酷的技术。天空是极限，任何都是可能的。这项技术将如何影响你的生活？它会影响商业吗？
4. 个人练习：回想最近一次某人告诉你一些不可能做到的事。是什么事？你听说过吗？为什么？

信息系统开发

本章概述和学习目标

本章在第1章的基础上进一步深入介绍系统开发过程。成功的系统开发由基本的原理支配着，这些原理将在本章介绍。我们也将介绍一种具有代表性的系统开发方法，作为开发信息系统应遵循的方法。虽然这样一种方法并不一定保证成功，但它仍将增大成功的机会。本章将介绍以下内容：

- 按照用于质量管理的能力成熟度模型（CMM）描述系统开发过程的动机。
- 区分系统生命周期和系统开发方法。
- 描述系统开发的10个基本原理。
- 定义问题、机会和指示——系统开发项目的源动力。
- 描述用于把问题、机会和指示进行分类的PIECES框架。
- 描述系统开发基本阶段。对于每个阶段，描述它的目的、输入和输出。
- 描述覆盖了多个系统开发阶段的跨生命周期活动。
- 描述贯穿系统开发基本阶段的几种典型“开发路线”。描述不同的项目应该如何组合或者定制开发路线。
- 描述用于系统开发的各种自动化工具。

本章关键术语

能力成熟度模型（Capability Maturity Model, CMM）是用来评估组织的信息系统开发以及管理过程和产品的成熟度等级的框架。它由5个开发成熟度等级构成。

系统生命周期（system life cycle）将一个信息系统的生命分为两个阶段：1）系统开发阶段；2）系统运行和支持阶段，首先构建系统；然后使用系统，运行系统并支持系统；最后，从运行和支持阶段再回到开发阶段。

系统开发方法（system development methodology）是一个十分正式且精确的系统开发过程，它为系统开发人员和项目管理者定义了（在CMM第3级）一组活动、方法、最佳实践、交付成果和自动化工具，用来开发和维护大部分或所有的信息系统和软件。它的一个同义词是系统开发过程。

项目管理（project management）是界定范围、规划、组织人员、组织、指导和控制一个项目的活动，在最低开销和规定的时间内，以可接受的质量开发信息系统。

成本效益（cost-effectiveness）是在开发、维护和运行信息系统的成本与从系统中获得的效益之间取得平衡。成本效益使用一种称为成本效益分析的技术。

信息系统战略规划（strategic information systems plan）是一个正式的战略规划（一般3~5年），用于构造和改进信息技术架构以及使用该架构的信息系统。

企业战略规划（strategic enterprise plan）是整个企业的战略规划（一般3~5年），定义企业的任务、愿景、目标、战略、基准和评价准则。企业战略规划一般由业务部门战略规划补充完善，定义每个部门如何为企业做出贡献。（以上的）信息系统战略规划是部门级规划之一。

逐步投入（creeping commitment）方法是在整个项目过程中都持续地重新评价可行性和风险，并相应地调整项目预算和最后期限。

风险管理（risk management）是及早发现项目中可能出现的错误，防止它真正威胁信息系统的实现，对它们进行评估和控制。风险管理由风险分析和风险评估所驱动。

问题（problem）是不期望发生的情况，它妨碍组织完整地实现其任务、愿景、目标和/或指示。

机会 (opportunity) 是指即使在没有出现具体问题的情况下, 也能改善组织的可能性。

指示 (directive) 是一个由管理层、政府或其他外部影响强加的新需求。

指导委员会 (steering committee) 是由系统所有者和信息技术主管组成的管理机构, 它为候选的系统开发项目排序, 批准相应项目。

积压项目 (backlog) 是由于优先级低于被批准的项目而无法获得经费支持或人员支持的项目建议。注意优先级是不断变化的, 所以, 积压项目在以后也可能获得批准。

问题陈述 (problem statement) 是对问题、机会和指示的描述和分类, 也可能包括约束条件和对解决方案的一个初始视图。同义词包括初始研究和可行性评估。

约束条件 (constraint) 是可能制约解决方案或问题解决过程的因素、限制条件或约束。

范围蔓延 (scope creep) 是一种常见现象, 其中项目的需求和预期经常不顾对预算和进度的影响而缓慢增加。

工作陈述 (statement of work) 是与管理层和用户团体之间的开发或提升某个信息系统的合约, 它定义目标、范围、约束条件、高级用户需求、进度和预算。同义词包括项目章程、项目计划和服务等级合约。

系统模型 (system model) 是一幅系统的图示, 表示现实情况或者希望的情况。系统模型促进了系统用户、系统分析员、系统设计人员和系统构造人员之间的交流。

逻辑设计 (logical design) 将业务用户需求转换成系统模型, 该模型仅仅描绘了业务需求, 而没有描述这些需求的任何可能的技术设计或实现。常见的同义词有: 概念设计和要点设计。后者是指建模系统的“要点”, 或者独立于任何技术的“基本需求”。逻辑设计的反义词是物理设计 (在本章后面定义)。

分析瘫痪症 (analysis paralysis) 是一个讽刺词汇, 描述了一种常见的项目状态, 这时过多的系统建模极大地减缓了实现系统方案的进度。

物理设计 (physical design) 将业务用户需求转换成系统模型, 描述用户的业务需求的技术实现。常见同义词包括: 技术设计或实现模型 (用于描述输出)。反义词是逻辑设计。

系统支持 (system support) 是对系统用户的不断的技术支持, 以及处理可能出现的错误、失误或新的需求所需的维护工作。

跨生命周期活动 (cross life-cycle activity) 是存在于系统开发方法中多个阶段或者所有阶段的活动。例如: 调查研究、记录文档、演示汇报、估计和度量、可行性分析、项目和过程管理、变化管理及质量管理。

调查研究 (fact-finding) 是一个正式过程, 使用调研、面谈、会议、调查表、抽样和其他技术收集关于系统、需求和喜好的信息。这个活动也称为信息收集或者数据收集。

记录文档 (documentation) 是记录一个系统的事实和规格说明以供现在和以后参考的活动。

演示汇报 (presentation) 是交流发现的情况、建议和文档的活动, 供感兴趣的用户和管理者评审。演示汇报可以是书面的, 也可以是口头的。

资料库 (repository) 是一个数据库和/或文件目录, 系统开发者在其中存储一个或多个信息系统或项目的所有文档、知识和产品。知识库通常是自动化的, 以便信息存储、访问和共享。

可行性 (feasibility) 是对组织将要开发的信息系统的价值或实用性的度量。

可行性分析 (feasibility analysis) 是度量和评估可行性的活动。

估计 (estimation) 是对系统开发所需的费用和努力值的预测值。一个不太恰当的同义词是猜测, 意思是估计基于实验或试验性证据, 但缺少严密性——即猜测。

过程管理 (process management)

瀑布开发方法 (waterfall development approach) 是一种系统分析和设计方法, 要求每个阶段必须在另一个阶段之后“完成”。

迭代开发方法 (iterative development approach) 是一种系统分析和设计方法, 它在一系列连续的迭代中

完成整个信息系统。每个迭代要进行足够的分析、设计和构造。同义词包括增量和迭代。

模型驱动开发（Model-Driven Development, MDD）技术强调绘制模型以可视化地分析问题、定义业务需求以及设计信息系统。

逻辑模型（logical model）是一种图示的表示方法，描述系统是“什么”或“做什么”。同义词包括本质模型、概念模型和业务模型。

物理模型（physical model）是一种技术图示的表示方法，描述系统是“什么”或“做什么”以及系统在技术上“如何实现”。同义词包括实现模型和技术模型。

过程建模（process modeling）是一种以过程为中心的技术，其最流行的形式是结构化分析和设计方法，使用描述业务过程需求的模型推导出有效的软件设计。结构化分析引入了一个称为数据流图的建模工具，用来说明数据通过一系列业务过程的流程。结构化设计将数据流图转变成一个称为结构图的过程模型，说明实现业务需求的自顶向下的软件结构。

数据建模（data modeling）是一种以数据为中心的技术，用来建模业务数据需求，以及设计实现这些需求的数据库系统。最常见的数据模型是实体关系图。有时也称为数据库建模。

对象建模（object modeling）技术试图在被称为对象的单一结构中融合数据和过程要素。对象模型从对象和对象之间关系的角度文档化系统。对象建模技术是面向对象分析和设计方法的基础。

快速应用开发（Rapid Application Development, RAD）是一种系统开发策略，该策略强调用户深入地参与到一系列系统工作原型的快速进化和构造过程中，以加速系统的开发过程，系统工作原型最终将成为目标系统（或者系统的一个版本）。

原型（prototype）是一个小规模、有代表性的或者可工作的模型，这个模型反映了信息系统的用户需求或建议设计。任何原型都可能会忽略某些功能或特征，直到原型最终完全进化成需求的一个可接受的实现系统为止。

时间盒（timeboxing）是一段不能延长的时间段（通常为 60 ~ 90 天），系统的第一个版本（或下一个版本）必须在这个时间段内投入运行。

商用应用软件包（commercial application package）是一种可以购买到并定制（在一定限度内）的软件应用，以满足大量组织或特定行业的业务需求。同义词是商用现成产品（COTS）系统。

建议申报书（request for proposal, RFP）是一种与软件供应商交流应用软件包的业务、技术和支持需求的正式文档，这些软件供应商希望竞争销售应用软件包或服务。

报价申报书（request for quotation, RFQ）是一种与单个软件供应商交流应用软件包的业务、技术和支持需求的正式文档，该软件供应商已经被选中提供应用软件包或服务。

差距分析（gap analysis）是将对商用软件包的业务和技术需求与特定商用软件包的功能和特征进行比较，以定义不能满足的需求。

计算机辅助软件工程（Computer-Assisted Software Engineering, CASE）使用支持系统模型的绘制和分析的自动化工具，有些 CASE 工具也提供原型设计和代码生成能力。

CASE 资料库（CASE repository）是一个系统开发人员的数据库。它是开发人员存储系统模型、详细描述和说明以及系统开发的其他产品的地方。资料库的同义词包括字典和百科全书。

正向工程（forward engineering）是 CASE 工具的一种能力，能够直接从系统模型生成初始软件或数据库代码。

逆向工程（reverse engineering）是 CASE 工具的一种能力，能够直接从软件或数据库代码生成初始的系统模型。

应用开发环境（Application Development Environments, ADE）是集成的软件开发工具，它提供了以最快的速度 and 最高质量开发新应用程序所需的全部工具。常用的同义词有集成开发环境（IDE）。

过程管理软件（process manager application）是一个自动化工具，它帮助记录文档与管理一套方法学和开发路线、交付成果以及质量管理标准。它的同义词是 methodware。

项目管理软件（project manager application）是一个自动化工具，它帮助规划系统开发活动（最好使用

认可的方法学)、估计和分配资源(包括人力和经费)、调度活动和资源、按照进度和预算监督进展、控制和修改进度和资源,以及报告项目进展。

2.1 系统开发过程

本章重点介绍信息系统开发,我们将分析系统开发过程(参见1.5节)。注意,我们没有说“这个”过程,因为有多少位专家,就会有多少种不同的开发过程。我们只介绍一种开发过程并在本书中一直使用。

如今,组织更是别无选择,只能采用并遵循某种系统开发过程!首先,使用一致的系统开发过程能够提高效率,管理层可以在项目之间调动资源;其次,一致的方法产生一致的文档,减少了维护系统的生命期费用;最后,美国政府已经强制任何寻求为政府开发软件或固件的组织,必须符合一定的质量管理要求。总之,一致的过程提高了质量。而且,为了增加竞争优势,许多组织已经积极地投身于全面质量管理之中。为了提高质量和生产率,许多组织转向类似于能力成熟度模型的质量框架。

2.1.1 能力成熟度模型

随着组织中标准信息系统开发过程的成熟,项目时间和费用在减少,同时生产率和质量在提高。卡内基·梅隆大学的软件工程研究所观察并度量了这种现象,开发出能力成熟度模型(CMM),以帮助所有的组织实现这些优点。CMM在业界和政府中有很多的追随者,基于CMM的软件评价被用作所有联邦政府项目合同的资质证明。

信息系统和软件的CMM框架用来帮助组织改善其系统开发过程的成熟度。CMM包括5个成熟度等级(见图2-1):

第1级——初始级:这一级有时称为无政府状态(anarchy)或混乱状态(chaos)。在这一级,系统开发项目没有规定的过程可遵循。文档是零散的,或者项目与项目之间的文档不一致,这给那些必须在系统生命期间维护系统的人提出了难题。几乎所有的组织一开始都处于第1级。

第2级——可重复级:组织已经建立了项目管理过程和实践来跟踪项目费用、进度和功能。这一级的重点是项目管理。组织总是采用某个系统开发过程,但项目与项目之间可能不同。

第3级——已定义级:组织购买或者开发了一个标准的系统开发过程(有时称为方法学)。由于所有项目都使用标准化的过程,所以每个项目都会产生

一致且高质量的文档和交付成果。开发过程是稳定的、可预测的,而且是可重复的。

第4级——已管理级:组织建立了可度量的质量和生产率目标。标准系统开发过程和产品质量的详细度量数据被例行公事地收集和存储在数据库中,并且组织根据收集的数据努力提高项目管理水平。这样,管理层寻求更主动地而不是被动地应对系统开发问题(例如费用超支、范围蔓延、进度延迟等)。

第5级——优化级:根据第4级建立的度量和分析,标准化的系统开发过程被连续地监督和改进。经验教训在整个组织内部共享,强调保证质量的同时消除系统开发过程中的低效率。

注意,CMM中的每个等级都是下一等级的先决条件。



图 2-1 能力成熟度模型 (CMM)

目前，许多组织正努力至少达到 CMM 第 3 级。达到第 3 级（已定义级）的核心是使用标准开发过程或方法学构建或集成系统。如表 2-1 所示，通过把 CMM 第 3 级的过程改进措施制度化，组织可以在进度和费用上实现显著的改进。^①

表 2-1 系统开发“过程”对质量的影响

一个 200 000 行代码开发项目的 CMM 项目统计						
组织的 CMM 等级	项目持续时间（月）	项目人月	出现的错误数	平均费用（百万美元）	最低费用（百万美元）	最高费用（百万美元）
1	30.0	600	61	5.5	1.8	100 +
2	18.5	143	12	1.3	0.96	1.7
3	15.0	80	7	0.728	0.518	0.933

资料来源：Master Systems 公司。

2.1.2 系统生命周期和系统开发方法

“系统生命周期”和“系统开发方法”这两个词经常被不恰当地混淆。大部分系统开发过程源自一个自然的系统生命周期，系统生命周期只是自然存在的。图 2-2 表示了生命周期的两个阶段，注意两个关键事件触发了两个阶段之间的转换：

- 当系统从开发阶段循环到运行和支持阶段时，必然发生一次转换。
- 在某个时刻，出现报废，系统将从运行阶段循环到重新开发。

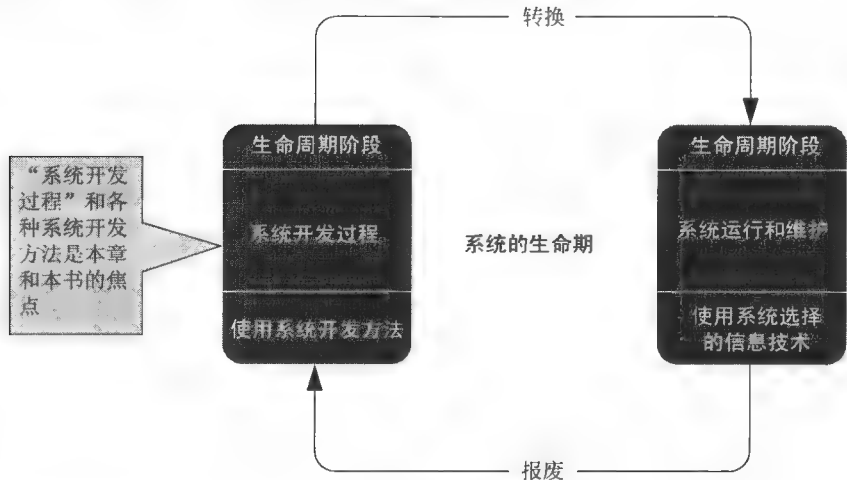


图 2-2 系统生命周期

实际上，一个系统在同一时刻可以处于多个阶段。例如，版本 1 可能处于系统运行和支持阶段，而版本 2 正处于系统开发阶段。

什么是系统开发方法呢？系统开发方法“执行”系统生命周期的开发阶段。每个信息系统都有它自己的生命周期。方法学是构建和维护系统以及所有其他信息系统走过它们的生命周期的标准过程。

方法可以购买，也可以自创。下面列出了一些系统开发方法的例子，包括：结构化快速应用开发（Architected RAD）、动态系统开发方法（DSDM）、联合应用开发（JAD）、信息工程（IE）、快速应用开发（RAD）、Rational 统一开发过程（RUP）、结构化分析和设计（虽然旧，但仍经常遇到）、极限编程（XP）。注意：存在许多基于以上一般方法的商用方法和软件工具（称为 methodware）。本书将讲授这些方法的基本原理。

① 公司白页：“Rapidly Improving Process Maturity: Moving Up the Capability Maturity Model through Outsourcing”（Boston: Keane, 1997, 1998, p. 11）。

2.1.3 系统开发基本原理

在开始介绍方法之前，我们首先介绍所有系统开发方法的一些基本原理。

2.1.3.1 原理1：让系统用户参与

应该把系统开发看作是系统用户、系统分析员、设计人员和构造人员之间的一次合作。系统分析员、设计人员和构造人员对系统开发负责，但他们必须抽出时间同所有者和用户交流，坚持请他们参与项目，并对可能影响他们的决策寻求所有关联人员的支持。

交流不畅和误解仍是系统开发中存在的主要问题。但是，所有者和用户的参与和培训可以减少这类问题的发生，并有助于新想法和技术被人们所接受。因为人们往往抵制变化，所以信息技术经常被视为是一种威胁。克服这种威胁的最好方法，就是与所有者和用户经常全面地交流。

2.1.3.2 原理2：使用一套问题解决步骤

系统开发方法首先是构建系统的问题解决步骤。问题这个词在本书中大量使用，其含义包括：1) 真正的问题；2) 改进的机会；3) 来自管理层的指示。系统分析员应该在所有项目中都使用这套问题解决步骤的某个变种。

缺少经验的问题解决者往往省去或者忽略以上步骤中的一步或几步。例如，可能没有完全理解问题，或者过早地选取了想到的第一种方案。结果会出现以下几种情况：1) 解决了错误的问题；2) 没有正确地解决问题；3) 挑选了错误的方案；4) 挑选了非最佳方案。当正确利用了系统开发方法的问题解决步骤时，应该能够减少或者消除以上风险。

2.1.3.3 原理3：确立开发阶段和开发活动

所有的生命周期方法都规定了相应的开发阶段和开发活动。不同的作者、专家以及公司对开发阶段和开发活动的数量和范围的描述都不尽相同。

这些阶段是：①范围定义、②问题分析、③需求分析、④逻辑设计、⑤决策分析、⑥物理设计和集成、⑦构造和测试、⑧安装和发布。这些阶段都将在本章后面讨论。这些阶段不是绝对顺序的，在开发阶段之间往往重叠，如图2-3所示。而且，开发阶段可以根据项目的特定需求进行裁剪（例如：最后期限、复杂性、战略、资源等）。本章将描述经过整套方法或问题解决过程的“开发路线”的每种定制。

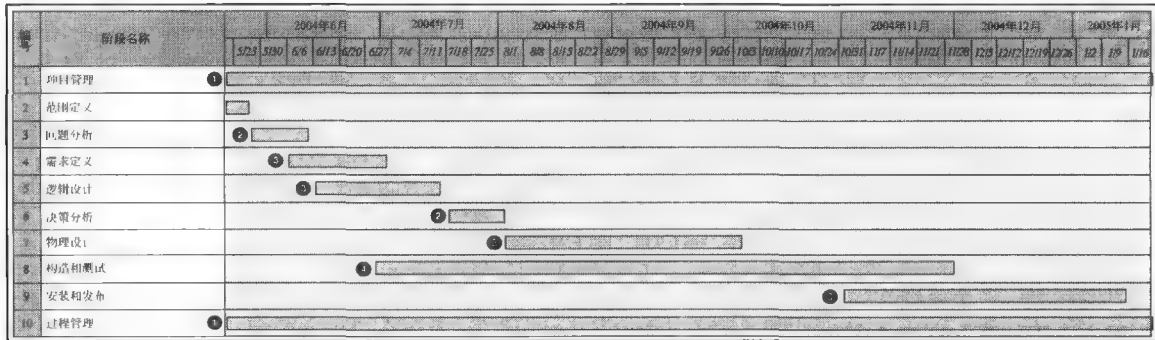


图 2-3 系统开发阶段的重叠

2.1.3.4 原理4：在开发过程中记录文档

在大中型组织中，系统所有者、用户、分析员、设计人员和构造人员不断变化，有些人被提升了，有些人退休了，有些人退出了，有些人又重新加入进来。为了促进不断变化的关联人员之间的有效交流，文档必须随同整个系统开发工作同时展开。

文档提高了多个关联人员之间的通信和相互接受程度，展示了系统的优点和缺陷，促进了用户参与度，并再次确保对进度的管理。但是，有些开发方法被批评要求过多的文档，以至于对过程或最终的系统没有增加多少价值。我们的方法在文档的价值和编写文档的工作量之间寻求平衡，专家们称之为敏捷建模。

2.1.3.5 原理5：建立标准

系统集成对于任何组织的信息系统的成功都很关键。为了实现或者改进系统集成，组织需要遵从标准。在许多组织中，这些标准以企业信息系统架构的形式出现。信息系统架构设置一些标准，指导技术解决方案和信息系统采用一个公共的技术愿景或者配置。

如果没有 IT 架构，每个信息系统和应用软件可能会使用完全不同的技术进行构建。这不仅使得集成应用软件变得困难，而且会带来资源管理问题——IT 组织不能根据情况变化（或者新出现的紧急情况）在项目之间方便地调整开发人员，因为不同的团队是由不同的技术专家构成的，他们基于不同的技术来开发信息系统。建立一个企业信息技术架构并推动项目和团队执行此架构意义重大。

当新技术出现时，IT 架构需要跟着改变，但改变是可以管理的。组织的技术主管（CTO）通常负责探索新技术并管理 IT 架构。给定了架构，所有的信息系统项目都必须遵循该架构实现新系统（除非获得 CTO 的特许）。

2.1.3.6 原理6：管理过程和项目

大多数组织都有一个系统开发过程，但在项目中的使用并不总是一致。不仅开发过程，而且需要管理开发过程的项目。过程管理确保一个组织选定的过程在所有项目中得到一致地运用。过程管理还定义和改进选定的过程（参见 1.6 节）。项目管理确保以最小的开销、在规定的时间内、以可接受的质量（使用标准的系统开发过程）开发信息系统。

过程管理和项目管理受质量管理的需求影响。过程中的质量标准确保了每个阶段的活动和交付产品对高质量的信息系统的开发做出贡献。它们降低了遗漏问题和需求以及有缺陷的设计和程序错误（bug）的概率。标准也使 IT 组织变得敏捷。由于每个项目都遵循同一个被共同理解和接受的过程，所以当发生人员变动时，组织可以在项目之间调整人员。

2.1.3.7 原理7：将信息系统作为重要的投资看待

信息系统是一种重要的投资，就像一个车队或者一个新的建筑一样。系统所有者对此投资负责。注意最初的责任出现在项目的早期，即当系统所有者同意启动项目时。之后（在称为决策分析阶段），系统所有者再次确认投入更多投资的决策。当考虑一种重要的投资时，有两个问题必须考虑。

1. 对于任何问题，都会有几种可能的解决方案。分析员（或用户）不必一定接受想到的第一种方案，看不到其他方案的分析员可能会给企业带来损害。
2. 在确定多种方案之后，系统分析员应该评价每种方案的可行性，特别是**成本效益**，成本效益通过成本效益分析技术度量。

就像项目管理和过程管理一样，成本效益分析在整个系统开发过程中都要进行。

分阶段的系统开发方法的一个明显优点是，它提供了几次重新评价成本效益、风险和可行性的机会。实际上，常常存在把项目进行下去的诱惑，其原因仅仅是因为已经投入了很多。从长远角度来看，取消项目比实现它的损失要小得多！对年轻的分析员来说，记住这一点十分重要。

2.1.3.8 原理8：不必害怕取消和返工

有句谚语说：“不要再在错事上投钱。”换句话说，无论已经投入多少经费，都不要害怕取消项目或缩小项目范围——减少你的损失。最后，我们推荐一种用于系统开发的**逐步投入方法**^①。使用逐步投入方法，系统开发方法建立了多个可行性检查点。在每个检查点上，所有的费用都被认为是过去的（意思是不可恢复的），所以它们同决策无关。这样，在每个检查点上都应该重新评价项目，以确定继续投入时间、精力和资源的计划是否仍可行。在每个检查点上，分析员应考虑以下方面的因素：

- 如果项目不再可行就取消它。
- 如果项目范围增加了，就需重新评价并调整费用和进度。
- 如果不能改变项目预算和进度，并且项目预算和进度不足以实现所有的项目目标，就减小范围。

① Thomas Gildersleeve, *Successful Data Processing Systems Analysis*, 2nd ed. (Englewood Cliffs, NJ: Prentice Hall, 1985), pp. 5-7.

财务分析员或多或少对成本泥潭这个概念有所了解，但大多数分析员、用户甚至系统所有者常常忘记它，从而越陷越深。

除了在整个项目过程中管理可行性，我们还需要管理风险。风险管理寻求风险和收益之间的平衡。不同的组织对风险的承受能力不同，意味着有些组织比其他组织愿意承担更大的风险，以获得更大的收益。

2.1.3.9 原理 9：分而治之

无论你们是否意识到，在整个学生生涯中，大家都在学习分而治之。从高中开始，老师就教你在写文章之前要先列提纲，提纲就是一种分而治之的写作方法。系统开发中也使用了类似的方法。为了更容易地解决问题并构建更大的系统，我们将一个系统分解成子系统和组件，在系统分析中，我们称之为分解。通过不断地将一个系统分解成更容易管理的小系统（子系统），分析员可以简化问题解决的过程。通过将系统的不同部分委托给不同的关联人员，这种分而治之的方法对沟通和项目管理也是一种有益的补充。

信息系统框架中的构件提供了分解信息系统开发过程的基础，我们将在本书中一直使用该框架。

2.1.3.10 原理 10：设计系统时应考虑到增长和变化

企业在不断地变化，业务需求在变，业务的优先次序在变。相应地，支持业务的信息系统也必须不断变化。因此，好的方法应该接受变化的现实。系统应该设计成能够适应增长和变化的需求。换句话说，设计优良的信息系统能够随着业务增长，并适应之。但无论我们设计的系统如何适应增长和变化，总有再也无法支持业务的时候。

系统科学家用熵来描述所有系统都随着时间不可避免地出现衰退的现象。正如本节前面所述，系统实现后，就进入了生命周期的运行和支持阶段。在这个阶段，分析员将遇到修改系统的需求，有的是改正简单的错误，有的是重新设计系统以适应变化的技术，还有的是进行修改以支持变化的用户需求，应有尽有。这种修改导致分析员和程序员重复生命周期中以前完成的阶段。最后，维护当前系统的费用超过了开发一个新系统的费用，目前的系统达到了熵值而作废。

我们已经说明了方法学的 10 个基本原理，可以用来评价任何一套方法学。

2.2 系统开发过程

本节将介绍一个系统开发的逻辑过程。我们将首先研究项目的类型以及如何启动项目；然后介绍 8 个开发阶段；最后，介绍为不同类型的项目和开发策略建立的贯穿各个开发阶段的不同开发路线。

2.2.1 项目确定

绝大部分项目由系统所有者和用户启动。大部分项目的推动力源自问题、机会和指示。为简化起见，我们将使用问题一词来总括地指代“问题、机会和指示”，因此，问题解决也包括了解决问题、探索机会和实现指示。

潜在问题的种类太多，以至于不能将其全部在本书中列出。但是，James Wetherbe 开发了一个实用的问题分类框架^①。这个框架称为 PIECES，其名称是由 6 类问题中每一类问题的首字母组合而成。这 6 类问题分别是：

- P——改进性能（performance）的需要。
- I——改进信息（information）（和数据）的需要。
- E——改进经济（economics）、控制成本或增加收益的需要。
- C——改进控制（control）或安全的需要。
- E——改进人与过程的效率（efficiency）的需要。
- S——改进对客户、供应商、合作伙伴、雇员等的服务（service）的需要。

① James Wetherbe and Nicholas P. Vitalari, *Systems Analysis and Design: Traditional, Best Practices*, 4th ed. (St. Paul, MN: West Publishing, 1994), pp. 196-199. James Wetherbe 是一个受人尊敬的教育家、研究人员和咨询顾问。

图 2-4 扩展了 PIECES 分类的内容。

PIECES 框架的分类既不是穷尽的，也不是互斥的——有重叠。任何一个给定的项目通常涉及一类或更多的类，并且任何给定的问题或机会都可能暗含在多个类中。PIECES 是一个实用的框架（用于 FAST），而不只是一个学术研究成果。后面我们还会探讨 PIECES。

项目可以是计划中的，也可以是非计划中的。

规模再大的信息服务组织也很容易被计划外项目所淹没，所以，项目常常由指导委员会检查，并排列先后次序。指导委员会由系统所有者和 IT 经理构成，决定哪个请求获得批准。那些没有被批准的项目请求被积压下来，直到资源许可（有时永远也不会发生）为止。

PIECES 问题解决框架和检查表

下面的检查表用来确定问题、机会和指示，它使用了 Wetherbe 的 PIECES 框架。注意，PIECES 中的类别不是互斥的，可能有些问题出现在多个列表中。而且，可能出现的问题也不能尽数。PIECES 框架同样适用于分析手工的和计算机化的系统和应用。

性能

- A. 吞吐量——在一些时间段内执行的工作量。
- B. 响应时间——事务或请求与响应之间的平均延迟。

信息（和数据）

- A. 输出
 - 1. 缺少某些信息。
 - 2. 缺少必要的信息。
 - 3. 缺少相关的信息。
 - 4. 太多的信息——“信息超载”。
 - 5. 无用格式的信息。
 - 6. 不正确的信息。
 - 7. 难以生成的信息。
 - 8. 不能及时提供给后续使用的信息。
- B. 输入
 - 1. 没有收集数据。
 - 2. 没有及时地收集有用的数据。
 - 3. 没有正确地收集数据——包含错误。
 - 4. 难以收集数据。
 - 5. 冗余地收集数据——同样的数据收集了多次。
 - 6. 收集了太多的数据。
 - 7. 收集了非法的数据。
- C. 数据存储
 - 1. 数据冗余地存储在多个文件和/或数据库中。
 - 2. 数据存储不正确。
 - 3. 数据由于事故或者故意破坏而不安全。
 - 4. 没有很好地组织数据。
 - 5. 数据不灵活——不易满足对数据存储的新需求。
 - 6. 数据不可访问。

经济

- A. 费用
 - 1. 费用不明确。
 - 2. 费用不可追踪到源头。
 - 3. 费用太高。
- B. 利润
 - 1. 可被开发的新市场。
 - 2. 可以改进的现有市场。

- 3. 可以增加的订单。

控制（和安全）

- A. 安全或控制太少
 - 1. 没有充分地编辑输入数据。
 - 2. 对数据实施（或可能实施）犯罪（例如：欺骗、窃取）。
 - 3. 违背道德——将数据和信息给了未经授权的人。
 - 4. 在不同的文件或数据库中冗余存储的数据相互不一致。
 - 5. 数据隐私法规或规定被侵犯，或者可能被侵犯。
 - 6. 出现处理错误（由人、机器或软件造成）。
 - 7. 出现决策错误。
- B. 安全或控制太多
 - 1. 官僚作风使系统变慢。
 - 2. 控制使客户或雇员感到不方便。
 - 3. 过多的控制引起处理延迟。

效率

- A. 人、机器或计算机浪费时间。
 - 1. 冗余地输入或复制数据。
 - 2. 冗余地处理数据。
 - 3. 冗余地生成信息。
- B. 人、机器或计算机浪费材料和供应。
- C. 任务所需的工作量太大。
- D. 任务所需的材料太多。

服务

- A. 系统产生了不正确的结果。
- B. 系统产生了不一致的结果。
- C. 系统产生了不可靠的结果。
- D. 系统不易学习。
- E. 系统不易使用。
- F. 系统难以使用。
- G. 系统对新情况或异常情况反应不灵活。
- H. 系统对变化反应不灵活。
- I. 系统与其他系统不兼容。

图 2-4 用于确定问题的 PIECES 框架

计划中的项目和计划外项目都遵循同样的基本系统开发过程。下面我们就详细地介绍项目开发的各个阶段。

2.2.2 项目开发阶段

大多数方法由多个开发阶段构成。开发阶段的数量在各种方法中不尽相同。第1章介绍了系统开发生命周期的4个经典阶段。我们在此介绍的方法采用8个阶段更详细地定义开发周期里程碑和发布物。因为我们要学习系统开发工具和技术，所以我们将在这本书中使用这种方法。

图2-5描述了我们的方法的典型开发阶段。每个阶段都产生可交付产品，并传递到下一个阶段。随着每个阶段的完成，文档不断累积。注意我们在图中包括了图标形式表示的构件，它们表示了系统开发过程中知识和过程产品的积累。另外，项目来自用户团体的“问题”、“机会”和“指示”开始，以提供给用户团体“可用的业务方案”结束。

图2-6从信息系统构件（从第1章和第2章已有所了解）的角度表示了我们的方法。阶段在图的右侧，其中，可交付产品围绕着知识、过程和通信构件构建，关联人员位于图的左侧。我们扩展并重复了一些关联人员的图示，以反映他们参与了多个阶段。

注意 本节以下内容简要描述这8个基本阶段。在整个介绍过程中，我们将参考图2-5的过程视图，以及图2-6的构件视图。另外，图2-5和图2-6中，所有以小写字母出现的词汇都表示阶段、前提条件（输入）和可交付产品（输出）。

2.2.2.1 范围定义阶段

范围定义阶段是一个典型的项目的第一个阶段。该阶段有两个目标。首先，它回答一个问题：“这个项目值得考虑吗？”其次，假定该项目值得考虑，它确定项目的范围、目标、约束和限制条件以及所需的项目参与者、预算和进度。

如图2-6所示，范围定义阶段的参与者主要包括“系统所有者”、“项目经理”和“系统分析员”。系统用户一般不包括在内，因为这个阶段对于用户最终将提供的细节程度来说太早了。

在图2-5中，我们看到范围定义阶段由一些“问题”、“机会”和“指示”的组合（其中会加入“约束条件”和“目标”）触发。该阶段有几个发布物。一个重要的输出是“问题陈述”，有关触发项目的问题、机会和/或指示的简要描述。PIECES框架为问题陈述提供了一个绝好的提纲。该阶段的目标不是解决这些问题、机会和指示，而是对它们进行分类。我们还应该确定可能影响项目的约束条件，例如：预算限制、最后期限、人力资源、企业政策或政府法规，以及技术标准。最后，系统所有者应该为他们期望的系统改进提供至少一个高级视图。

假定我们已经对问题、机会、指示、约束条件和目标设想有了基本的了解，我们就需要建立初始范围。因此，初始的“范围陈述”是该阶段的另一个重要输出。范围定义了这个项目有多大。信息系统构件为定义范围提供了一个有用的框架。在图2-6中，范围和设想使用“信息”、“功能”和“接口”这些词汇定义。在项目进行过程中，范围可能（通常也确实会）会变化。但通过记录初始范围，我们就为控制（预算和进度的）范围蔓延建立了一个基线。

给定项目的初始问题和范围陈述，分析员就可以组织项目团队，估计系统开发的预算，并为后续阶段准备一个进度安排。最终，该阶段以系统所有者的“是否继续”决策结束。系统所有者或者同意建议的项目范围、预算和进度，或者必须缩小范围（以减少开支和时间），或者取消该项目。在图2-5中，这个可行性检查点用一个菱形表示。

最终也是最重要的发布物是“工作陈述”。工作陈述是开发信息系统的合约或协议，它合并了问题陈述、范围陈述以及进度和预算，提供给所有将参与该项目的各方。

2.2.2.2 问题分析阶段

一般来说，不管其目前是否使用计算机，总存在一个现有系统。“问题分析”阶段研究现有系统，分析发现的问题，促使项目团队更深入地理解引发该项目的问题。分析员经常会发现新问题，并回答一个最重要的问题：“解决这些问题的收益是否超过了构建系统的开销？”

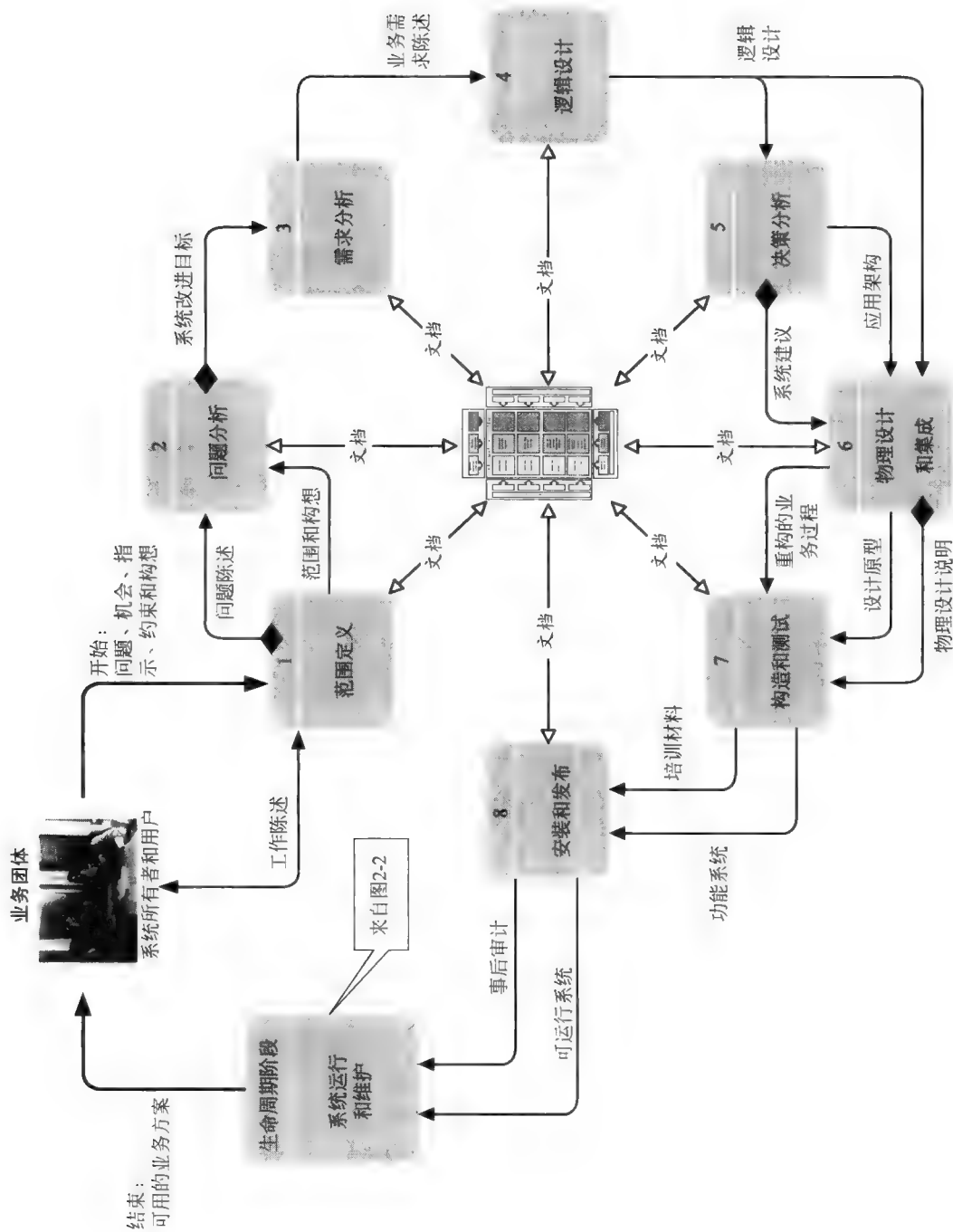


图 2-5 系统开发的过程视图

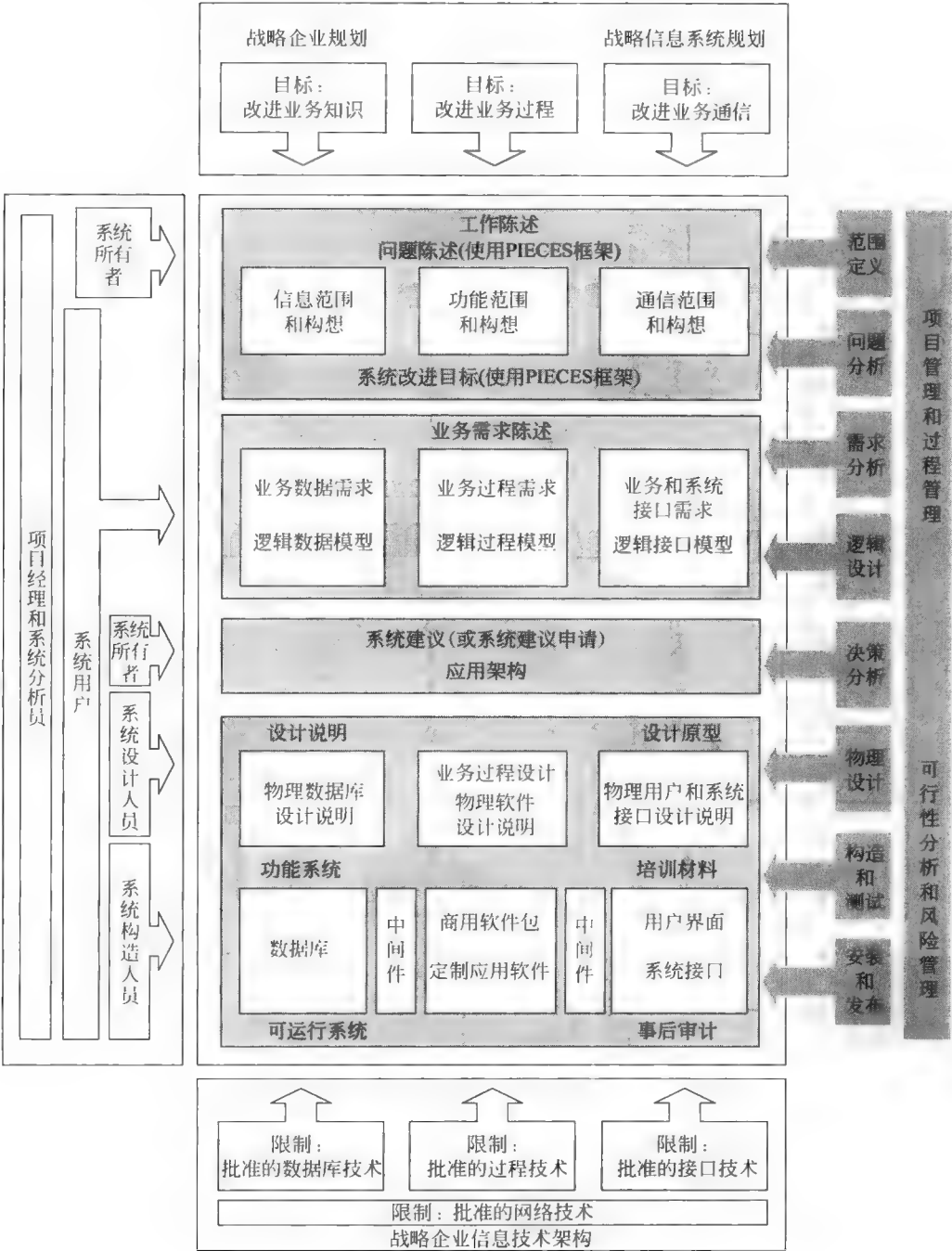


图 2-6 系统开发的构件视图

图 2-6 再次提供了问题分析阶段的信息系统构件形式的图示。注意参与者仍包括“系统所有者”，但这个阶段开始主动地包括“系统用户”。在任何项目中，系统用户都是业务领域的专家（注意图中有意地将系统用户的视角与多个阶段重叠——原理 1：“让系统用户参与”）。当然，“项目经理”和“系统分析员”一直参与项目的所有阶段。

如图 2-5 所示，问题分析阶段的前提条件是范围定义阶段定义和批准的“范围和问题陈述”，发布物是一组“系统改进目标”，来自对业务问题的全面理解。可以将系统改进目标考虑成评价任何（最终

设计和实现的)新系统的升级准则。系统改进目标可以向系统所有者和用户做口头汇报,也可以进行正式的书面报告。

每个现有系统都有自己的术语、历史、文化和微妙之处。了解这些内容是这个阶段的重要副产品。从收集到的所有信息中,项目团队更好地理解现有系统的问题和机会。检查这些调查结果之后,系统所有者要么同意问题分析阶段的建议,要么不同意。为了同逐步投入的原理相一致,我们在问题分析阶段的最后设了另一个“继续或停止”可行性检查点(菱形)。这个项目可能会:

- 如果认为问题不值得解决,则取消该项目。
- 同意继续进入下一阶段。
- 削减或者扩大范围(同时修改预算和进度),然后同意继续进入下一阶段。

2.2.2.3 需求分析阶段

假定系统所有者同意继续下去,那么我们就可以设计一个新系统,是这样吗?并非如此!新系统应该为用户提供什么功能?必须收集什么数据?存储什么数据?期望达到什么样的性能等级?注意,也就是要求做出关于系统必须做什么(而不是如何做这些事情)的决策。“需求分析阶段”就是要定义业务需求,并且为它们排序。简单地说,分析员同用户接触,找出用户需要什么,或者他们想从新系统中得到什么,并要小心地避免讨论任何技术或技术实现。需求分析阶段也许是系统开发中最重要的阶段。需求分析中的错误和疏忽将导致用户对最终系统不满意,也会导致费用高昂的修改。

再次参考图2-6,注意该阶段的参与者主要包括“系统用户”(其中可能包括将实际使用系统的系统所有者)和“系统分析员”。“项目经理”也要参与。这个阶段没有“系统设计人员”,以便防止过早地关注技术方案。构件本身就可以作为定义业务需求的框架,其中包括“业务数据需求”、“业务过程需求”以及“业务和系统接口需求”。因为业务需求的目标是解决问题,PIECES也可以为需求陈述提供一个有用的提纲。

如图2-5所示,来自问题分析阶段的“系统改进目标”是需求分析阶段的前提,交付物是“业务需求陈述”。再次说明,需求陈述不要规定任何技术方案。需求陈述可以是一份几页纸的文档,也可以是每个需求准备一份几页纸的文档。

为了写出业务需求陈述,系统分析员与系统用户密切合作,确定需求及其优先级。这些信息通过面谈、调查表以及组织会议收集,对团队的挑战在于验证这些需求。系统改进目标是业务需求的“试金石”——是否每个需求都满足一个或几个系统改进目标?第6章和第7章将介绍用来确定和记录用户需求的系统分析工具和技术。

通常,需求也必须排列优先级,这有两个作用。首先,如果项目时间要求变得紧迫,需求优先级可被用来重新确定项目范围;其次,优先级经常用来定义设计和构造系统的迭代层次,以形成最终产品的分级发布版本。

绝不应该省去或者简化需求分析阶段。对新系统和应用的一个最常见的抱怨就是它们没有真正满足用户需求。当系统设计人员和构造人员在没有完全理解业务需求之前就过早地进入技术方案时,经常会出现这种情况。在应用具体技术之前,系统设计人员和构造人员需要依靠出色的系统分析员与用户一起定义完整、正确的业务需求。

2.2.2.4 逻辑设计阶段

业务需求通常是口头表达的。系统分析员要把这些话语转换成系统模型图,以验证需求的完整性和一致性(图2-5是一种常见的系统模型[数据流图]的例子)。系统建模印证了一个永恒的概念:“一幅图胜过千言万语”。

“逻辑设计阶段”将业务需求转换成系统模型。逻辑设计这个词意味着“与技术无关”,即模型图独立于任何可能的技术方案展示了整个系统。因此,模型图对我们可能会考虑的任何技术方案都需要满足的业务需求进行了建模。

不同的方法对系统建模或逻辑设计的要求程度不同。传统的方法(结构化分析和设计、信息工程和 Rational 统一过程[RUP])通常要求绘制不同详细程度的多类/多个系统模型图。计算机自动工具可

以辅助系统分析员完成绘图工作。相反,敏捷方法(框架化快速应用开发和极限编程)建议“刚好足够的建模”。这些敏捷建模方法希望防止项目得分析瘫痪症。本书倾向于敏捷方法,但也认识到复杂的问题使用传统的方法更合适。

如图 2-6 所示,该阶段的参与者包括“系统分析员”(他们绘制模型)和“系统用户”(他们验证模型)。“项目经理”确保建模工作符合标准,并且没有延误项目进度。我们可以绘制:1)“逻辑数据模型”描述数据和信息需求;2)“逻辑过程模型”描述业务过程需求;3)“逻辑接口模型”描述业务和系统接口需求^①。

如图 2-5 所示,逻辑设计的前提是前面阶段的“业务需求陈述”。实际上,需求分析和逻辑设计阶段总是有大量的重叠。换句话说,当业务需求被确定并记录下来时,就可以进行建模。逻辑设计的交付物是“逻辑设计模型和规格说明”。根据采用的方法不同,规格说明的详细程度不同。例如,我们可以定义一条业务规则说明某个数据属性的合法值(如信用度),或者定义一条规则说明信用检查的业务策略。

2.2.2.5 决策分析阶段

当获得业务需求和逻辑系统模型之后,通常有大量的可选方案用来设计满足这些需求的新系统。

决策分析阶段的目的是:1)确定候选技术方案;2)分析这些候选方案的可行性;3)推荐一个候选系统作为目标方案进行设计。

在图 2-6 中,决策分析阶段位于整个开发过程的中间,一半的构件位于上方,另一半的构件位于下方。这同决策分析阶段作为从分析到设计的过渡阶段的角色是一致的,从“系统用户”到“系统设计人员”(最终到“系统构造人员”)。设计人员(专门技术领域的技术专家)开始与系统用户和“系统分析员”一同工作。分析员定义和分析可选方案。做出需要考虑使用的技术的决策,这些技术是技术架构的一部分。最终,“系统所有者”要做出批准或不批准的决策,因为他们要为项目付费。

如图 2-5 所示,验证过的业务需求加上逻辑系统模型和对需求的规格说明触发了决策分析阶段。项目团队从不同的人那里(为技术设计和实现)征求想法和观点,这其中也可能包括 IT 软件厂商代表的想法和观点。候选方案确定下来,并按照各种标准进行分类。许多现代组织具有信息技术和架构标准,这些标准限制了可以被考虑和分析的候选方案的数量(图 3-6 的信息系统构件模型的底部说明了这类标准的存在)。确定候选方案之后,每个候选方案都按照以下准则进行评价:

- 技术可行性——该方案在技术上是否能够实现?职员是否具有相应的技术专业知识来设计和实现这个方案?
- 运行可行性——该方案是否满足了用户需求?在多大程度上满足?该方案将会怎样改变用户的工作环境?用户对这个方案的感觉怎么样?
- 经济可行性——该方案是否划算(按照本章前面的定义)?
- 进度可行性——该方案能够在可接受的时间内设计和实现吗?
- 风险可行性——这一技术和方法成功实现的可能性有多大?

项目团队通常要找出最可行的方案,即提供技术可行性、运行可行性、经济可行性、进度可行性和风险可行性最佳组合的方案。不同的候选方案可能在某个方面最可行,但总有一个方案在所有方面最可行。

决策分析阶段的主要交付成果是“系统方案建议”。这个建议可以是书面的,也可以是口头的。可能有个输出。决策分析阶段的逐步投入检查点(图中菱形)可能导致发生以下情况:

- 同意并资助系统的设计和构造建议(如果范围明显地扩大了,还需要增加预算并调整时间表)。
- 同意或者资助其他候选方案中的某个方案。
- 驳回所有候选方案,要么取消该项目,要么返回重做新的方案提议。

① 你已经熟悉了面向对象建模的知识,应该注意到对象模型倾向于多少模糊框架的边界,但是框架仍然可以应用,因为待解决问题仍由框架图中的三个基本业务目标所驱动。这些内容将在本书的面向对象分析和设计章节介绍。

- 同意建议方案的一个范围缩小了的版本。

决策分析阶段也可能会为批准的方案生成一个“应用架构”，这个模型是推荐的或者批准的方案的高级蓝图（类似于一个简单的建筑平面图）。

在继续讲解之前，请注意在图 2-6 中有一个“系统方案建议”的变种，称为“系统建议请求”（或 RFP）。这是用作购买硬件和/或软件方案的建议。我们将在讨论商业软件集成过程时再进一步介绍这部分内容。

2.2.2.6 物理设计和集成阶段

当决策分析阶段批准了“系统方案建议”之后，就可以最终设计新系统。“物理设计和集成阶段”的目的是将（部分由“逻辑系统模型”表示的）业务需求陈述转变成用于指导系统构造的“物理设计说明”。换句话说，物理设计解决如何将技术应用于新系统中。设计将受到上一阶段批准的“应用模型”的制约，同时要符合内部技术设计标准，以确保完整性、可用性、可靠性、性能和质量。

物理设计与逻辑设计相对。逻辑设计主要处理独立于任何技术方案的业务需求，物理设计代表了某个特定的方案。图 2-6 从信息系统构件的角度表示了物理设计阶段，注意设计阶段主要关心系统的技术视图：1) “物理数据设计说明”；2) “物理业务过程”和“软件设计说明”；3) “物理用户和系统接口说明”。“系统设计人员”和“系统分析员”（可能是同一个人的不同角色）是主要的参与者，但是，设计的某些方面通常也需要同“系统用户”共享（例如屏幕设计和工作流）。在程序设计或数据库课程中，读者可能已经接触过物理设计说明。

存在两种物理设计的极端情况：

- 按规格说明设计——设计产生物理系统模型和详细的规格说明，作为构造用的书面（或计算机生成的）蓝图。
- 按原型设计——构造不完整但可工作的应用程序或子系统（称为原型），并基于用户和其他设计人员的反馈细化原型。

实际上，通常执行的是这两种极端的某种组合。

没有哪个新信息系统能够在不与组织中已有的信息系统打交道的情况下独立存在。因此，系统设计总是包括集成设计。新系统必须同其他信息系统集成，也要同企业自身集成。集成通常反映在物理设计模型和设计规格说明中。

图 2-5 显示了物理设计和集成阶段的交付物是“物理设计模型和规格说明”、“设计原型”和“重新设计的业务过程”的组合。注意我们为该项目设定了最后一个“继续或停止”可行性检查点（见图中菱形）。项目很少会在设计阶段后被取消，除非它在预算方面已经毫无希望或者远远落后于进度，但是可以减少项目实现范围以便在规定的时间内产生一个可接受的最小产品。或者进度表被延长，以便在多个版本中构建一个更完整的系统。需要调整项目计划（进度和预算），以反映这些决策。

现代方法学表现出一种合并设计阶段和构造阶段的趋势，即设计阶段和构造阶段经常相互重叠。

2.2.2.7 构造和测试阶段

给定“物理设计模型和设计说明（或设计原理）”后，我们可以为这个设计构造并测试系统组件。图 2-5 中“构造和测试阶段”的主要交付物是“功能系统”。构造和测试阶段有两个目的：1) 创建并测试一个实现业务需求和设计说明的系统；2) 实现新系统和现有系统之间的接口。另外，要生成“最终文档”（例如：帮助系统、培训手册、帮助中心支持、生产控制指令），准备用于培训和系统运行。构造阶段还可能包括购买软件的安装。

图 2-6 的信息系统框架为构造阶段确定了相关的构件和开发活动。其中的焦点位于构件的最后一行。项目团队必须构造或安装：

- 数据库——数据库可能包括支持日常业务事务的联机事务处理（OLTP）数据库、支持日常报告和查询的运行数据库（ODS），以及支持数据分析和决策支持的数据仓库。
- 商业软件包和/或定制软件——根据需要安装或者定制软件。应用程序按照上一个阶段的物理设计和/或原型构造应用程序。软件包和定制软件都需要经过全面的测试。

- 用户界面和系统接口——用户界面（如 Windows 和 Web 界面）需要构造并测试其可用性和稳定性。系统间接口也需要使用应用集成技术实现。注意中间件（一种系统软件）经常用于集成独立的数据库、软件和接口技术。我们将在本书的设计章节讲述更多的中间件技术。

这个阶段的参与者有“系统构造人员”、“系统分析员”、“系统用户”和“项目经理”（见图 2-6）。“系统设计人员”也可能参与其中，以澄清设计说明。组件一旦构造出来，一般都向用户演示，以征求反馈意见。

构造阶段中的一项最重要工作是对单个系统组件以及整个系统进行测试。一旦测试完成，这个系统（或者系统的一个版本）就做好了“安装和发布”的准备。

2.2.2.8 安装和发布阶段

下面做什么？新系统往往在一定程度上与企业目前的行为方式相背离，所以，分析员必须提供从老系统到新系统的平稳转换，并帮助用户应对正常的启动问题。因此，安装和发布阶段发布产品投入运行（有时称为生产）。

来自构造和测试阶段的功能系统是安装和发布阶段的主要输入（见图 2-5），交付物是“运行系统”。“系统构造人员”将系统从其开发环境安装到运行环境中。分析员还必须培训用户，编写各种用户手册和生产控制手册，将现有文件和数据库转换到新数据库中，并进行最后的测试。出现任何问题都可能回到前面已经完成的阶段重新工作。随着新问题的出现，系统用户提供不断的反馈。安装和发布阶段基本上考虑与构造阶段同样的构件。

为了提供向新系统的平稳转换，系统分析员应该准备一个转换计划。这个计划可以采取一种突然切换的形式，也就是在某个特定日期，老系统中止并被新系统代替，或者老系统和新系统同时运行，直至新系统被认为可以代替老系统为止。

安装和发布阶段还包括培训使用最终系统的人，以及为系统用户编写帮助文档。实现阶段通常包括某种形式的“事后审查”，度量完成的项目是否成功，这项活动是一项 TQM 工作，它对未来系统项目的成功是有贡献的。

2.2.2.9 运行和维护阶段

一旦系统投入运行，就需要对其生命周期的剩余阶段提供不断的系统支持。系统支持包括以下活动：

- 辅助用户——无论用户培训得多好，用户文档写得多么完善，最终用户还是会要求额外的辅助——产生了未预料到的问题、加入了新用户等。
- 修正软件缺陷——软件缺陷是在软件测试中没有发现而被漏掉的错误。软件缺陷是不可避免的，只要有良好的知识支持，但大多数情况能够解决。
- 恢复系统——人为错误或者硬件或软件失效可能会引起系统失效，而系统失效可能导致程序“崩溃”和/或数据丢失。于是系统分析员或技术支持专家可能会被叫去恢复系统——也就是恢复系统文件和数据库并重新启动系统。
- 调整系统适应新需求——新需求可能包括新的业务问题、新的业务需求、新的技术问题或新的技术需求。

最后，我们认为用户反馈、出现的问题或者业务需求的变化指出是该需要重新设计系统了。换句话说，系统到了退役期，应该启动一个新的项目来创建一个新的系统开发过程。

2.2.3 跨生命周期活动

系统开发还包括一些跨生命周期活动。这些活动在图 2-5 中没有特定画出，但它们对项目的成功至关重要。下面就简单地介绍这些活动。

2.2.3.1 调查研究

在项目开发过程中需要做许多调查研究。调查研究对于一个项目的早期阶段至关重要。在这些阶段中，项目团队了解一个企业的语汇、问题、机会、约束、需求和任务的紧急程度。调查研究也可以用在决策分析阶段、设计阶段、构造和测试阶段及安装发布阶段，但用得较少。在后面这些阶段中，

项目团队研究技术方案并征求有关技术设计、技术标准和工作组件的反馈意见。

2.2.3.2 记录文档和演示汇报

沟通技能是一个项目能否成功的基本要素。事实上，沟通技能差常常被认为是项目延期或返工的原因。有两种沟通技术在系统开发项目中很常见，即记录文档和演示汇报。

在所有的项目阶段中都有机会进行记录文档和演示汇报工作。在图2-7中，标注①的箭头代表了各种文档，标注②的箭头代表了经常需要演示汇报的地方，标注③的箭头代表了系统开发的文档和其他制品存储在一个资料库中。知识库保存文档，供需要时复用或返工。

2.2.3.3 可行性分析

同我们用于系统开发的逐步投入方法相一致，可行性分析是一个跨生命周期活动。方法的不同阶段可以应用不同的可行性度量，这些度量包括技术可行性、运行可行性、经济可行性、进度可行性和风险可行性（类似我们在决策分析阶段中介绍的那样）。可行性分析需要优秀的估计技术。

2.2.3.4 项目管理和过程管理

CMM将系统开发看作是一个必须逐项目进行管理的过程。由于这个和其他一些原因，项目和过程管理是持续不断的跨生命周期活动。管理的两种类型在前面已介绍，这里再重述一下其定义。过程管理定义了每个项目使用的方法——可以理解成盖楼的方法，来构建系统。项目管理则是把管理方法应用于单个项目时的实例。

系统开发中失败和取得部分成功的项目远远多于特别成功的项目。原因是许多系统分析员不熟悉系统开发工具和技术，或者没有接受过这方面的训练。但大多数失败是由于领导和管理不善，从而导致了无法完成或无法确定需求、费用超支以及推迟交付。

2.2.4 顺序开发和迭代开发

以上几节对开发阶段的讨论可能会使读者得出结论认为系统开发是一个自然的顺序过程，按照一个方向从一个阶段到另一个方向。这种顺序开发事实上只是一种方法，这种方法如图2-8a所示。为了简化起见，在图中我们使用了4个经典阶段，而不是8个阶段。这种策略要求每个阶段在另一个阶段之后“完成”，直到信息系统终结。在现实中，各个阶段可以在时间上一定程度地相互重叠。例如，系统设计部分可以先于系统分析完成之前开始。这种方法的可视化显示类似瀑布，通常被称为瀑布开发方法。

现代的信息系统分析员已经不再喜欢瀑布方法了。一种更流行的策略（如图2-8b所示），通常被称为迭代开发方法或者增量开发过程。这种方法要求完成足够的分析、设计和实现以便能够完全开发新系统的一部分并将其尽快投入运行。一旦系统的这个“版本”实现了，该策略就进行另外的分析、设计和实现，以便发布系统的下一个版本。这些迭代将持续下去，直到整个信息系统的所有部分都实现了。这种迭代和增量过程之所以流行，可以简单解释如下：系统所有者和用户长期以来一直抱怨使用瀑布方法开发和实现信息系统所需的时间太长。迭代和增量方法可以使我们定期地以更短时间段发布可用的版本。这提高了客户（系统所有者和用户）的满意度。

2.3 选择开发路线和策略

到目前为止，我们已经描述了构成方法的基本开发阶段。过去，一套方法对大多数项目都是适用的。但如今，项目类型、技术和开发策略太多了，一种方法再也不能适应所有的项目了！像许多现代方法一样，FAST提供了贯穿各个开发阶段的不同开发路线，以适应不同的项目类型、技术目标、开发者技能和开发策略。

本节将描述几个开发路线和策略。图2-9显示了开发方法策略的分类，请注意图中以下内容。

- 开发方法学和开发路线既可以支持内部构造软件方案，也可以支持从软件供应商处购买商用软件方案。一般来说，许多同样的方法和技术都可以用于二者。
- 方法学既可以是非常规范的（“涉及所有内容；遵循所有规则。”），也可以是相对灵活的（“依据某种指南根据需要变化”）。

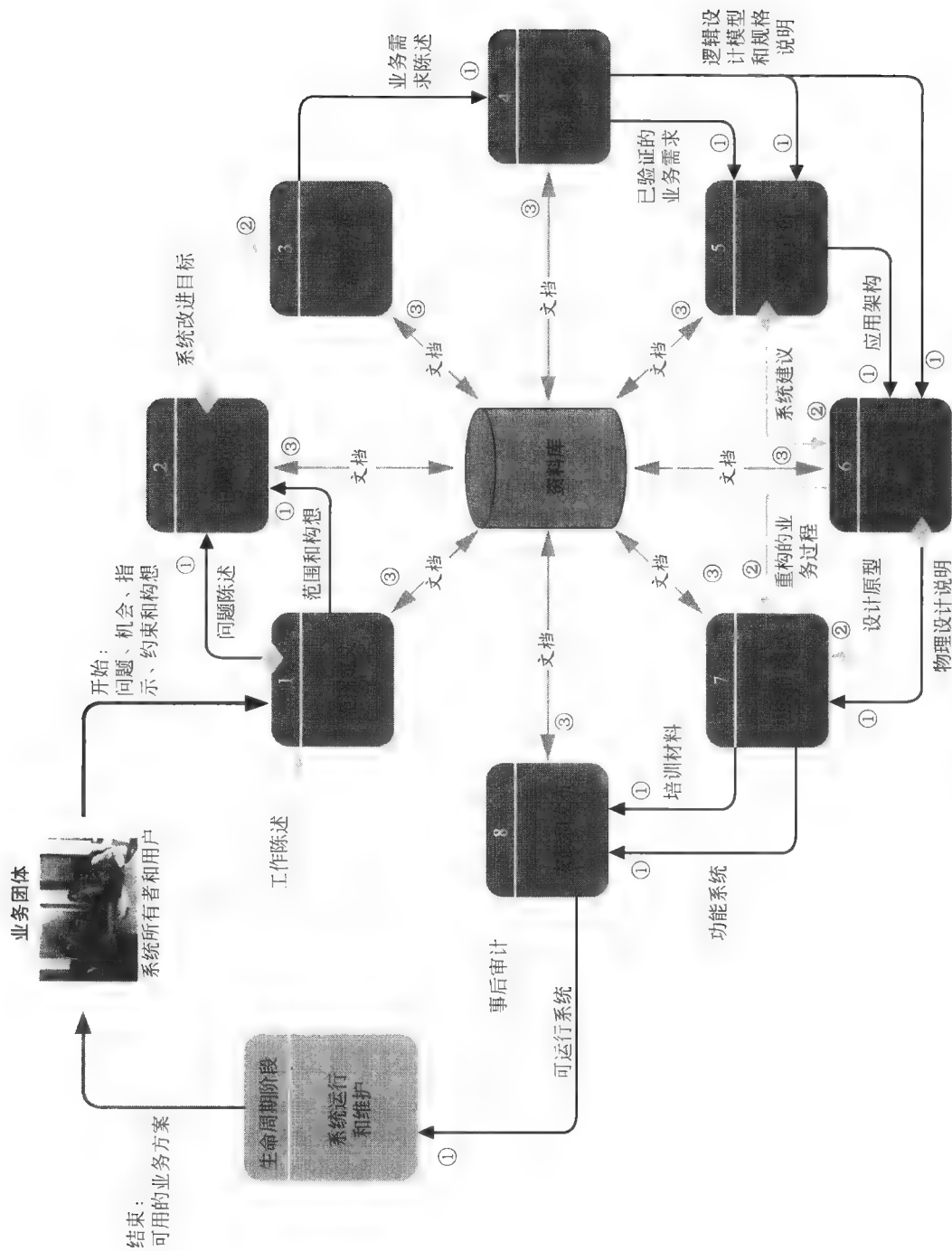


图 2-7 系统开发文档、资料库和演示

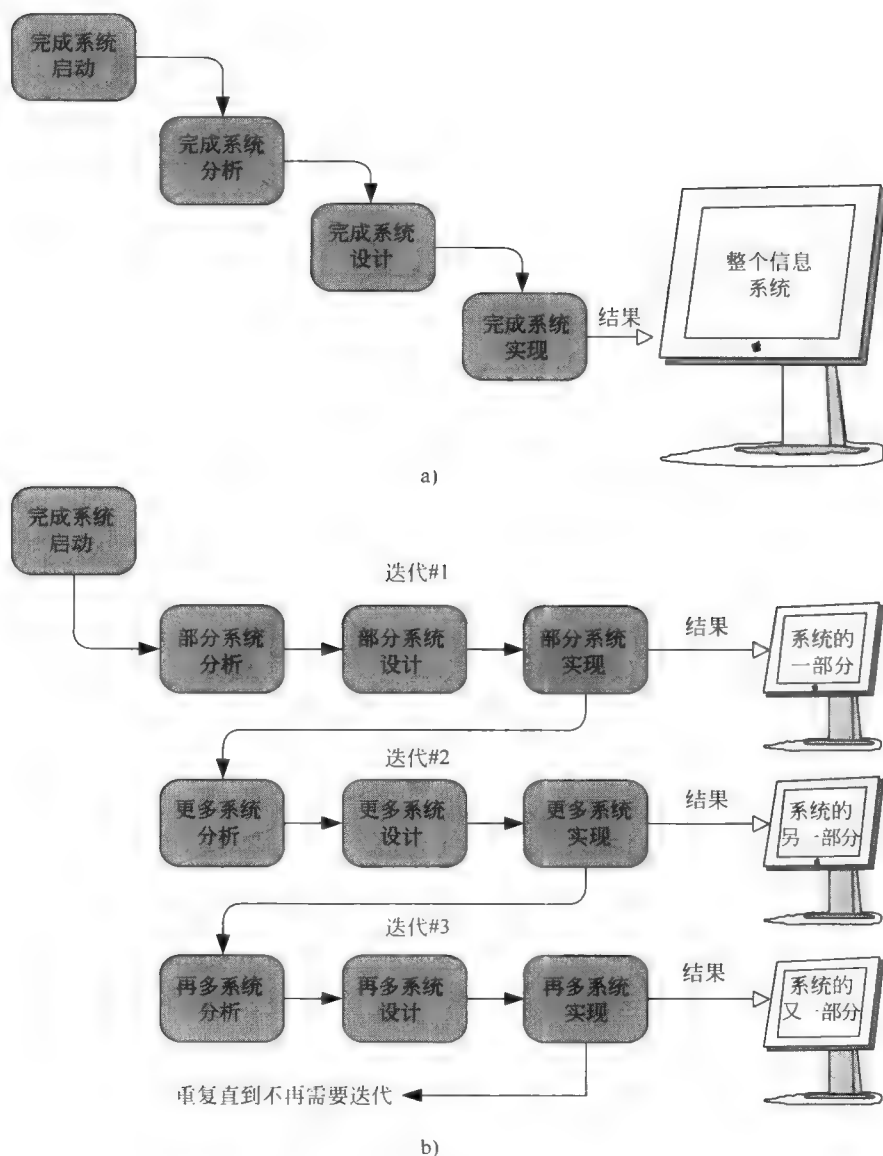


图 2-8 顺序开发方法与迭代开发方法
a) 顺序或“瀑布”策略 b) 迭代或增量策略

- 方法学既可以是模型驱动的（“绘制系统的图示”），也可以是产品驱动的（“构建产品然后观察用户的反应”）。
- 模型驱动的方法学正快速地发展成以面向对象技术为中心的，这是当今构造大多数系统使用的方法。早期的模型驱动的方法只强调过程建模或者数据建模。
- 最后，产品驱动的方法学趋于强调快速原型，或者尽可能快地编写程序代码（也许读者已经听说过极限编程）。

如此众多的策略！我们应该如何选择呢？当我们写作本书时，一种称为敏捷方法的运动正在形成。简单地说，敏捷方法的鼓吹者建议系统分析员和程序员掌握一个方法工具箱，其中包括来自所有以上方法学的工具和技术。他们应该根据问题和情况选择工具和技术。我们在本书中采用的 8 阶段方法是一种敏捷方法学。该方法学鼓吹集成使用来自许多方法的工具和技术，应用于可重复的过程上下文中（CMM 第 3 级）。

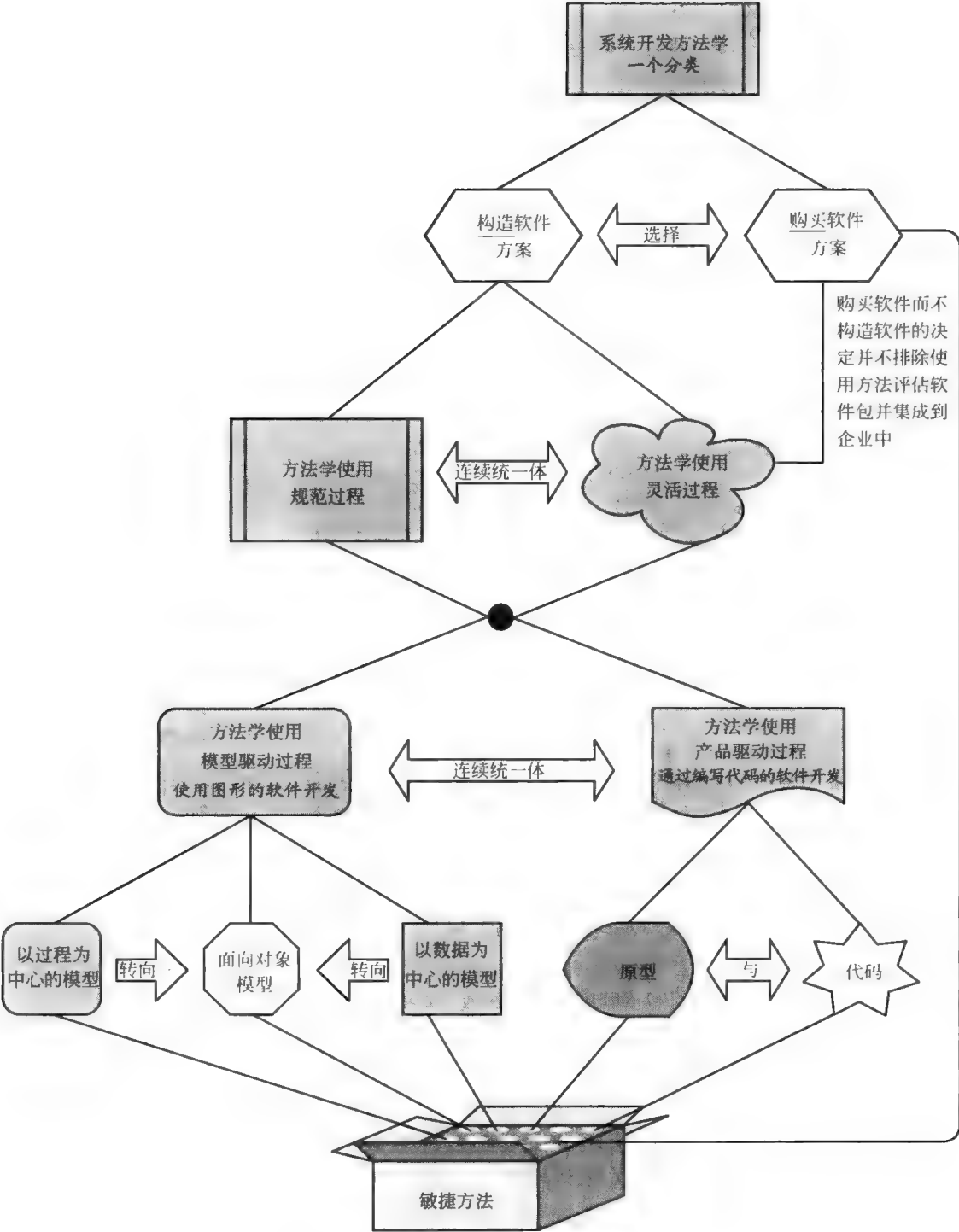


图 2-9 系统开发方法学和策略的分类

2.3.1 模型驱动开发策略

一个历史最悠久的、也是最常用的分析和设计信息系统的方法是建模。模型是系统的一种图形表示，表示了现实情况或者希望的情况。系统建模促进了系统用户、系统分析员、系统设计人员和系统构造人员之间的交流。在我们的方法中，系统模型用来说明和交流信息系统的“知识”、“过程”和“接口”构件。这种方法称为**模型驱动开发**。

模型驱动开发路线如图 2-10 所示。模型驱动方法同我们前面描述的基本阶段没有太大差别。请注意以下的注释，它们与图 2-10 数字注释相对应。

- ① 系统模型可能存在于创建当前系统的项目中。要小心，这些模型以过时而出名！但它们仍可以作为一个研究的起点来使用。
- ② 定义项目的范围很重要。交流项目范围的一种最简单的方法就是绘制“显示范围定义的模型”。范围模型显示了问题的哪些方面在范围之内，哪些在范围之外。这有时也称为上下文图。
- ③ 某些系统建模技术提倡“对现有系统详细地建模”，以确定系统问题和改进机会。这有时称为 as-is 系统模型。如今，建模现有系统已经显得没有意义了，许多项目经理和分析员认为这样只会产生反作用，或者没有什么价值。唯一的例外是建模现有业务过程，以便进行业务过程重构。
- ④ 需求陈述是系统开发中最重要的交付物之一。需求陈述有时包括“描述高层业务需求的模型”。如今一种最流行的建模技术是用例（见第 6 章）。用例确定需求，并在整个生命周期根据需求实现。
- ⑤ 大多数模型驱动技术要求分析员用**逻辑模型**记录业务需求。业务需求通常在“描述更详细的用户需求的模型”中描述。这些逻辑模型仅仅展示了系统必须是“什么”，或者必须“做什么”。它们与实现无关，也就是说，它们描述系统，而不考虑任何可能的技术实现方式。所以，它们适合描述和验证业务需求。
- ⑥ 作为决策分析阶段的结果之一，分析员可能会产生“描述应用架构的系统模型”。这类模型展示了系统的计划中的技术实现。
- ⑦ 许多模型驱动技术要求分析员开发“描述物理设计说明的模型”。**物理模型**不仅展示了系统是什么或者做什么，而且展示了系统在技术上如何实现。它们是与实现相关的，因为它们反映了技术选择和所选技术的限制。例子包括：数据库模式、结构图和流程图。它们是构造新系统的蓝图。
- ⑧ 新的信息系统必须与组织的业务过程交织在一起，相应地，分析员和用户可能会开发“重构的业务过程模型”。
- ⑨ 构造阶段将物理系统模型转化成软件。在某些情况下，自动化工具自动地将软件转换成“描述被构造的软件的物理模型”。这称为**逆向工程**。
- ⑩ 最后，运行系统可能包括“描述流程和规程的模型”，例如：系统模型可以记录备份和恢复程序。

综上所述，系统模型是绝大部分阶段交付成果的一部分。模型驱动的方法强调系统建模。模型一旦实现了，就成了生命周期的运行和支持阶段所需的任何改变的记录文档。

有几种不同的模型驱动技术，它们的差别主要是要求分析员绘制和验证的模型的类型不同。下面我们简要地介绍本书中将讲授的三种最流行的模型驱动开发技术。请注意我们在这里只是介绍这些技术，而不是介绍具体模型，我们将在后续章节中讲授这些模型。

2.3.1.1 过程建模

过程建模是在 1978 年的结构化分析和设计方法中提出的。虽然结构化分析和设计作为一种方法学已经不再流行，但过程建模仍是一种可靠而且重要的技术。信息系统构件包括几个视角：“知识”、“过程”和“接口”。过程建模主要关注“过程”构件。流程图就是一种过程模型（主要由系统构造人员使用），读者可能在程序设计课程中已经接触过了。由于业务过程重构的出现，过程建模又有所复兴。

数据流图和结构图有助于消除通常存在于非技术性的系统所有者和用户与技术性的系统设计人员和构造人员之间的交流隔阂。本书将讲授过程建模技术。

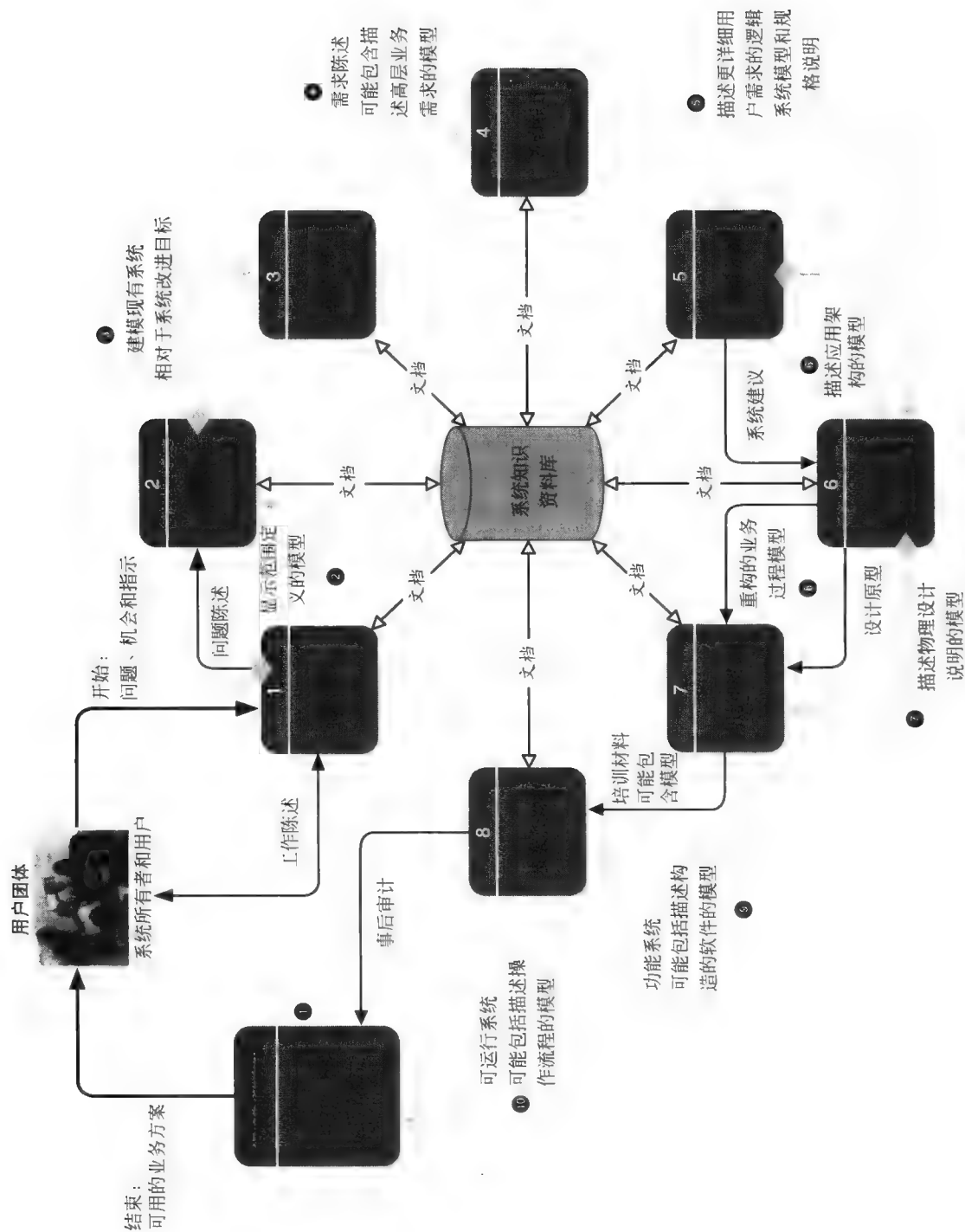


图2-10 模型驱动开发路线

2.3.1.2 数据建模

增进“知识”是信息系统框架的基本目标和基本构件。知识是信息的产物，信息又是数据的产物。数据建模方法强调知识构件，特别强调数据。在数据建模方法中，重点放在了捕捉业务数据需求的模型上，并把这些模型转换成数据库设计。数据建模是使用最广的系统建模技术，因此，数据建模技术也在本书中讲授。

2.3.1.3 对象建模

对象建模技术是技术进步的产物。如今，绝大多数编程语言和方法都是基于对象技术的。对象技术的概念将在本书中全面地介绍，这里只做一个简要介绍。

在过去30年，过程建模技术和数据建模技术有意将数据和过程分别加以考虑。换句话说，数据模型和过程模型是独立的。由于几乎所有的系统都既包括过程又包括数据，所以它们经常并行地使用，而且模型需要仔细地同步。对象技术是消除这种分离的考虑的一次尝试。这导致了对象建模方法的产生。

业务对象可以对应到企业中重要的真实事物，例如客户、客户订购产品的订单。每个对象既包括描述该对象的数据，也包括用来创建、读取、修改和删除该对象的过程。面向对象分析和设计（OOAD）极大地改变了信息系统构件的内容。“数据”和“过程”列（有时还包括“接口”列）本质上合并成了单一的“对象”列。模型专注于确定对象、构建对象以及将合适的对象装配成有用的信息系统。

2.3.2 快速应用开发策略

为了响应经济快速发展的步伐，快速应用开发（RAD）已成了一种加速系统开发的流行开发路线。RAD的基本思想如下：

- 让系统用户更主动地参与到分析、设计和构造活动中来。
- 将系统开发组织成一系列重点突出的研讨会，研讨会要让系统所有者、用户、分析员、设计人员和构造人员一同参与。
- 通过一种迭代的构造方法加速需求分析和设计阶段。
- 用户提前看到一个可工作系统的时间。

隐藏在构造原型之后的基本原理是：当用户看到系统工作时，他们才知道想要的是什么。在RAD中，原型最后会进化成最终的信息系统。快速应用开发路线如图2-11所示，以下的注释对应图中的数字标注：

- ①重点是缩短开发应用软件和系统的时间，所以初始问题分析阶段、需求分析阶段和决策分析阶段被合并并且被加速。简化了各个阶段的交付物，同样是因为对时间的考虑。交付物被称为是“初始的”，意思是随着项目的进展“可能会改变”。
在以上的初始分析阶段之后，RAD采用了迭代方法，如前所述。每次迭代的重点只是在几个星期内可以实现的足够多的新功能。
- ②“逻辑和物理设计说明”通常被极大地简化和加速。在循环的每次迭代中，仅仅考虑“某些”设计说明。虽然也会绘制一些系统模型，但这是有选择的，而且重点仍在快速开发上。其中的假设若是错误，可以在下一次迭代中被发现并被修改。
- ③在某些（但很少是全部）迭代中，有些业务过程可能需要重构以反映可能的软件应用集成。
- ④在循环的每次迭代中，都要构造和测试“一些设计原型或部分功能系统”。最终，循环的最后一次迭代将得到完成的系统。
- ⑤在每个设计原型或部分功能系统被构造和测试后，系统用户有机会“试用”原型。这样做是希望用户为下一次RAD循环迭代澄清需求、确定新的需求，并提供关于设计的“业务反馈”（如：易学习性、易用性）。
- ⑥在每个设计原型或部分功能系统被构造和测试后，系统分析员和设计人员将检查应用架构和设计，为下一次RAD循环迭代提供“技术反馈”和指导。

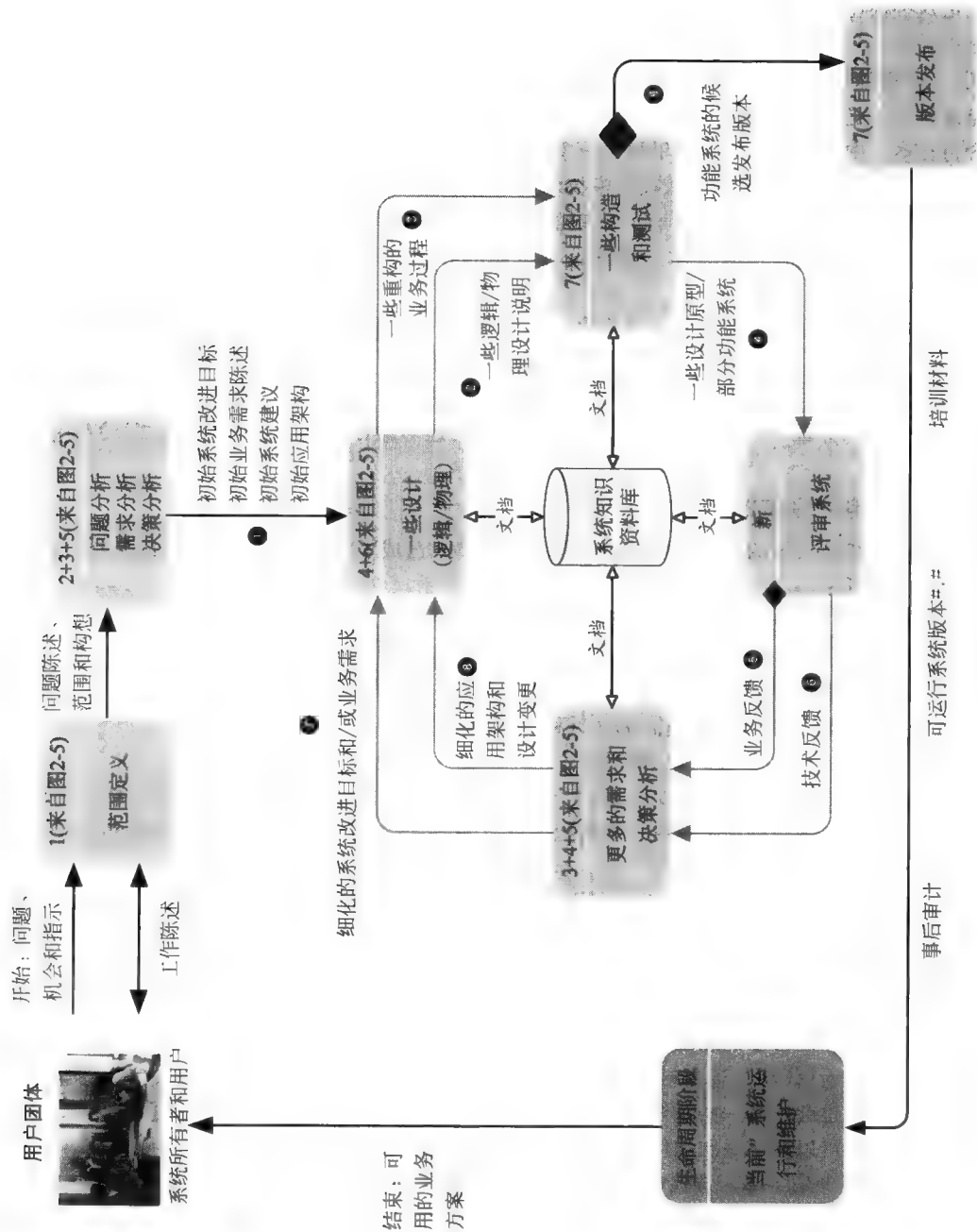


图2-11 快速应用开发 (RAD) 路线

⑦根据反馈信息，系统分析员将确定“优化的系统改进目标”和/或“业务需求”。这种分析的重点是修正或扩展目标 and 需求，并确定用户对设计的看法。

⑧根据反馈信息，系统分析员和系统设计人员将确定“优化的应用架构”和/或“设计修改”。

⑨最终，将认为系统（或系统的一个版本）是可以实现的。这个“功能系统的候选发布版本”经过系统测试并投入运行使用。系统的下一个版本可能正继续经过 RAD 循环迭代。

虽然不是对 RAD 的严格要求，但是原型循环的持续时间可以使用一种称为时间盒的技术进行控制。时间盒技术试图有规律地向用户和管理层发布运行系统。时间盒技术的支持者认为该技术可以提高和保持管理者和用户对一个项目的热情，因为系统的工作版本按照一个有规律的模式发布。

RAD 在小型和中型系统项目中最流行。因此，我们将在本书相应的章节中讲述原型技术和 RAD 技术。

2.3.3 商用应用软件包实现策略

有时，购买一个信息系统比内部开发系统更有意义。事实上，许多组织只有在能够获得竞争优势的情况下才在内部开发软件。而且，对于许多核心应用软件（例如：人力资源、财务、采购、生产和分销）来说，构造的系统不能带来什么竞争价值，因而购买**商用应用软件包**。相应地，我们的方法学也包括一个商用应用软件包路线。

最终的商用软件方案是企业资源规划，或 ERP（见第 1 章的定义）。ERP 方案为整个企业提供了所有的核心信息系统功能。对大多数企业来说，ERP 实现往往是该组织曾经有过的最大的单一信息系统项目。它可能要花费上千万美元，并且需要一支由管理者、用户、分析员、技术专家、程序员和咨询顾问组成的团队。

用于商用软件集成的我们的方法学路线并不是真正针对 ERP 项目的。大多数 ERP 供应商提供了各自的方法学（和咨询伙伴），帮助客户实现这样一个大型软件方案。相反，我们的方法学为企业可能购买的其他类型信息系统提供了一个开发路线。例如，一个企业可以为单个业务功能购买一个商用软件包，例如财务、人力资源或采购。该软件包必须被选择、安装、定制和集成到企业中以及其他现有的信息系统中。

商用软件包路线如图 2-12 所示，下面的注释对应图中的数字标注：

- ①需要注意购买软件包的决定在问题分析阶段做出。菱形表示“构造/购买”决定。以下的讨论假定已经决定购买。
- ②问题分析阶段通常包括一些初始的“技术市场调研”，以确定存在哪些软件包方案、这些软件有什么特征，以及用来评价这些应用软件的准则。这种调研可能会涉及软件供应商、IT 研究服务机构（如 Gartner Group）或者咨询顾问。
- ③定义了业务需求之后，必须同提供候选的软件包的供应商进行交流。业务需求以**建议申报书**（RFP）或**报价申报书**（RFQ）的形式组织并发送给候选的软件供应商征求意见。双向箭头表示可能需要对需求和评价准则进行一些澄清。
- ④供应商提交他们的应用程序方案的建议或报价。这些建议方案按照 RFP 中确定的业务和技术需求进行评价。双向箭头表示声明的特性和功能必须进行验证，有时还需要澄清。这些工作在决策分析阶段进行。
- ⑤同获胜的供应商协商软件合同和订单，以及安装和维护软件可能需要的服务合同。
- ⑥供应商提供基本软件和文档，软件的安装和实现服务通常由供应商提供，或者由软件的服务提供商（认证的咨询顾问）提供。
- ⑦当购买了应用软件包后，几乎所有组织都需要改变它的业务过程，以便更有效地利用该软件。业务过程重构不受欢迎，但通常又是必需的。许多情况下，要改变的东西并没有错——它们只是与软件不一样。系统用户对于改变他们日常做事的方式会感到不舒服。
- ⑧很少有应用软件包能够在安装时满足所有的业务需求。通常，需要进行**差距分析**以确定软件包的功能和特征不能满足哪些业务需求。

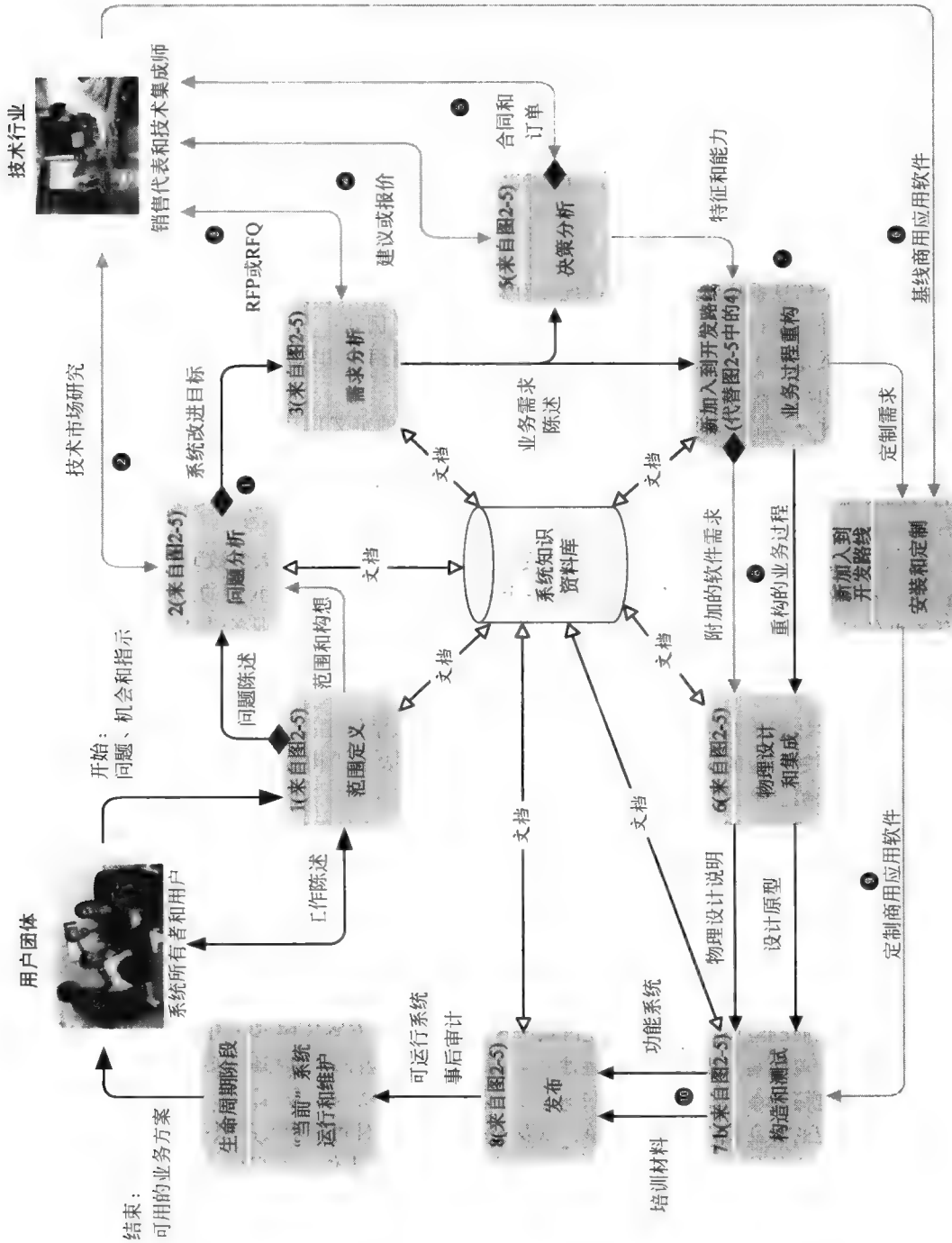


图 2-12 商用软件包开发策略

- ⑨ 安装并测试了基本软件，基于选项、偏好和参数的允许的定制也完成并经过了测试。注意：这些定制在软件供应商规定的限度内一般会在版本升级中保持。大多数情况下，软件供应商都是提供这种层次的定制商用软件。
- ⑩ 设计并构造了所有的附加（新增）软件以满足附加的业务需求。系统最终经过测试并投入运行，使用基本 FAST 过程中同样的活动。

无论如何，购买商用软件的趋势不能忽视。许多企业要求在启动任何类型的内部开发项目之前首先考虑购买软件方案。因此，我们将在本书合适的章节讲述在这种环境下使用的工具和技术。

2.3.4 混合策略

开发路线并不是互斥的。任何一个项目都可以选择或者要求使用多条开发路线或开发路线变种的组合。使用什么开发路线总是在范围定义阶段期间进行选择，并作为工作陈述的一部分进行协商。一种常用的混合了模型驱动开发路线和快速应用开发路线的策略是增量开发策略。图 2-13 显示了增量开发策略的一种可能的实现。项目通过 4 个阶段将信息系统投入运行，每个阶段都采用 RAD 路线实现最终系统的一个版本。其他的开发路线变种也是可能的。

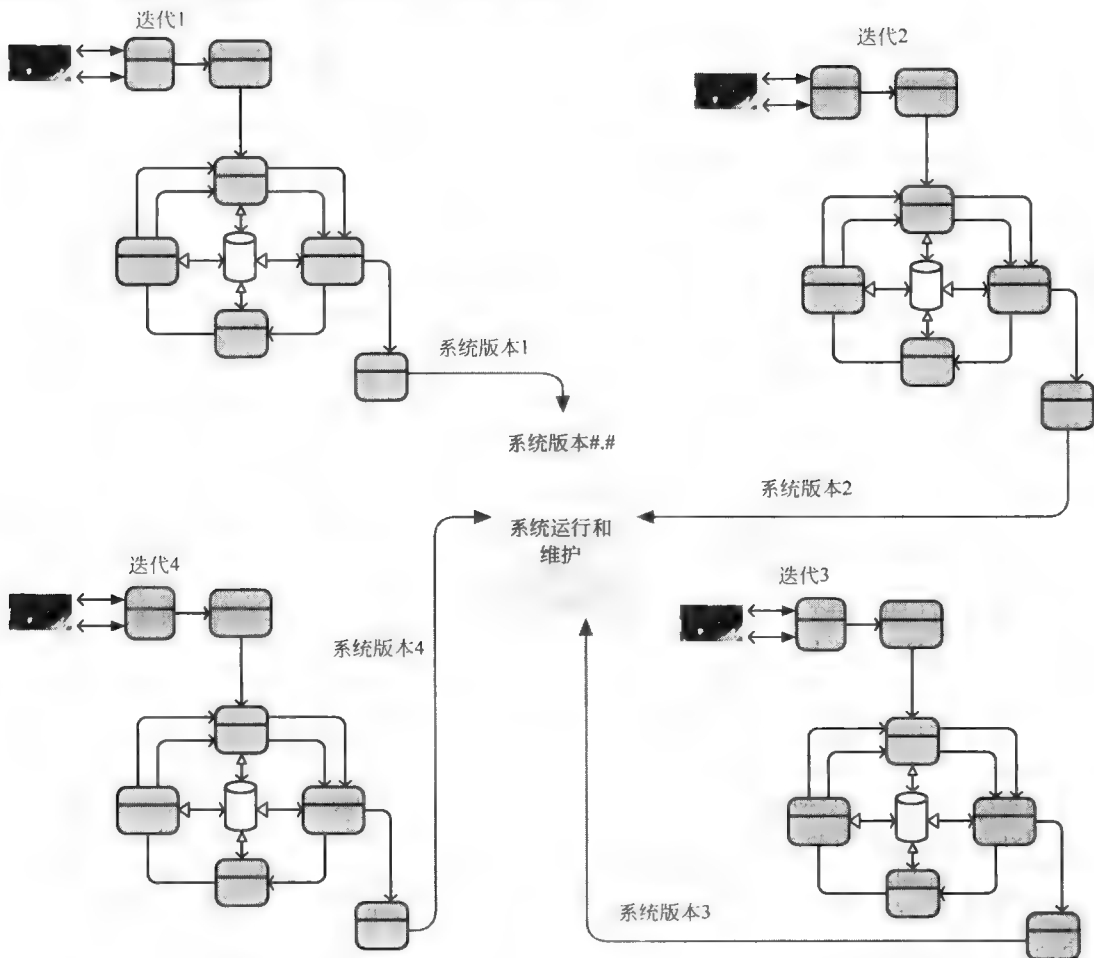


图 2-13 增量实现策略

2.3.5 系统维护

所有的开发路线最终都导致一个新系统投入运行。系统维护用于指引项目通过生命周期的运行和支持阶段——这可能会持续很多年！图 2-14 描述了系统维护的工作。系统维护不是一个独立的开发路

线,如图 2-14 所示,它仅仅是方法学应用于原先开发系统的小规模版本。系统维护的起点依赖于要解决的问题。请注意图 2-14 中以下数字标注:

- ①维护和再工程项目由用户反馈和技术反馈触发,这类反馈确定了新的问题、机会或指示。
- ②维护项目由“系统修改请求”启动,系统修改请求指出问题、机会或指示。
- ③最简单的修补是更正软件故障(错误)。这种项目一般直接跳到“重构和重测试”阶段,并较快地解决问题。
- ④有时,在系统实现后,一个设计缺陷才变得很明显。例如,由于令人混淆的界面设计,用户可能经常犯同样的错误。对于这类维护项目,将需要重新执行“物理设计和集成”阶段,之后是构造和发布阶段。
- 有时,“业务过程问题”变得很明显。这时,只是业务过程需要重构。
- ⑤偶尔,“新的技术需求”可能要求进行修改。例如,某个组织可能标准化了某个数据库管理系统(例如 SQL Server 或 Oracle)的最新版本。对于这类再工程项目将需要重新执行决策分析阶段,决定把一个现有的运行数据库转换到新版本的风险和可行性。这个项目最终将根据需要完成物理设计阶段、构造阶段和发布阶段。
- ⑥企业经常发生变化,所以,系统的业务需求也会随之变化。触发再工程项目的常见原因是新的业务需求或者要求改变的业务需求。给定需求后,必须重新执行需求分析阶段,重点关注新需求对现有系统的影响。根据需求分析,项目将继续执行逻辑设计阶段、决策分析阶段、物理设计阶段、构造阶段和发布阶段。
- ⑦随着企业的变化,可能会出现重要的新问题、机会和约束条件。在这类项目中,工作将从问题分析阶段开始,继续所需的后续阶段。
- ⑧在所有情况下,任何类型的维护和再工程项目的最后结果都是一个修改了的运行业务系统,它为系统用户和系统所有者提供改进后的价值。修改可能包括修订的程序、数据库、接口或业务过程。

系统开发方法学和开发路线至此就介绍完了。下面,我们将介绍支持现代系统开发的自动化工具在系统开发中的角色。

2.4 自动化工具和技术

你可能听说过一个有关鞋匠的寓言故事:鞋匠的孩子没鞋穿。这种情况很像有些系统开发人员面临的情况。多年来,我们使用信息技术来解决用户的业务问题,但是,我们将同样的技术应用到开发信息系统本身的程度却很低。不久以前,系统分析员的主要工具仍是纸、笔和流程图模板。

如今,整套的自动化工具已经被开发出来、推向市场并被安装使用,以辅助系统开发人员。虽然系统开发方法学并不总是需要自动化工具,但是大部分方法学都能从这些工具技术中获益。一些最常被引证的优点如下:

- 生产率提高——通过任务自动化。
- 质量改进——因为自动化工具检查了完整性、一致性和矛盾冲突。
- 更好、更一致的文档——因为工具使得编写和汇集一致且高质量的文档更加方便。
- 减少了生命期的维护——由于系统质量的提高以及产生了更优质的文档。
- 真正可用的方法学——通过使用强制性规则和内建的专业知识得以实现。

下面我们就简单地介绍每类自动化工具。

2.4.1 计算机辅助系统工程

系统开发人员长期以来一直期望能将信息系统和软件开发转变成类似工程的学科。系统工程和软件工程这些词汇都基于一个基本构想:系统开发和软件开发也应该能够按照工程一样的精确性和严格性实施。这种精确性和严格性同模型驱动的系统开发方法是一致的。为了帮助系统分析员更好地进行系统建模,业界开发了称为计算机辅助软件工程(CASE)的自动化工具。CASE 可以看作是用来设计

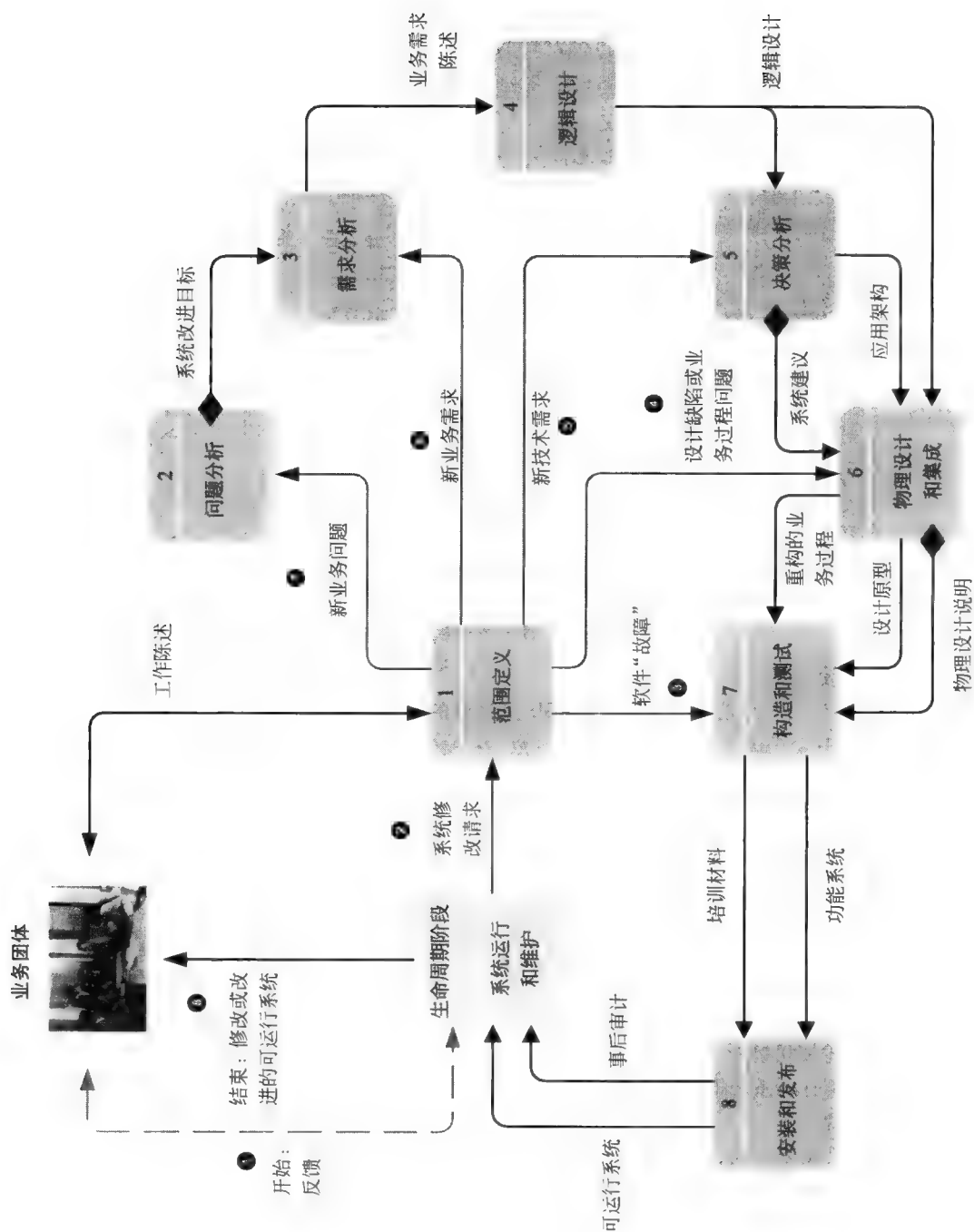


图 2-14 系统维护

和构造其他软件的软件，这同计算机辅助设计（CAD）技术十分类似。大多数现代工程师使用计算机辅助设计技术来设计其他产品，例如车辆、结构、机器等。有代表性的 CASE 工具包括：Computer Associates 的 Erwin、Oracle 公司的 Designer 2000、Popkin 公司的 System Architect、Rational 公司的 ROSE、Visible Systems 公司的 Visible Analyst。大部分 CASE 产品在个人计算机上使用，如图 2-15 所示。

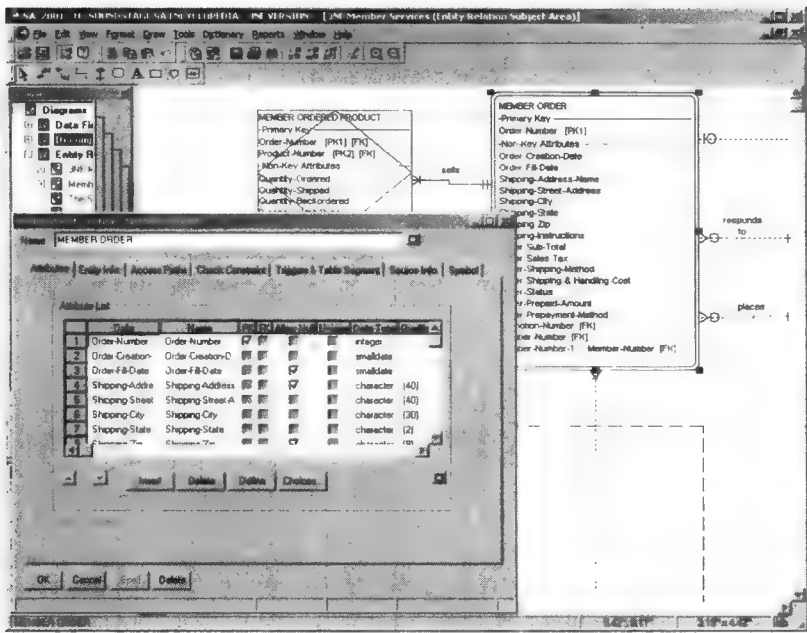


图 2-15 CASE 工具系统架构的屏幕捕捉

2.4.1.1 CASE 资料库

实际的 CASE 工具的核心是一个被称为 CASE 资料库的开发人员的数据库。资料库的概念在前面介绍过了（见图 2-7）。

围绕着 CASE 资料库的是构造系统模型和文档记录的工具集合。

2.4.1.2 CASE 工具

为了使用资料库，CASE 工具包提供了如下的一些工具，如图 2-16 所示。

- 作图工具用来绘制大多数系统开发方法学中要求或推荐的系统模型。通常，一个系统模型的形状可以链接到其他系统模型和详细描述。
- 字典工具用来记录、删除、编辑和输出详细文档记录和规格说明。描述可以同出现在系统模型中的形状关联起来，这些形状使用作图工具绘制。
- 设计工具可以用来构造系统构件，例如输入和输出。这些输入和输出可以与前面描述的系统模型及系统描述关联起来。
- 质量管理工具分析系统模型、系统描述、规格说明以及设计的完整性、一致性和是否符合方法学可接受的规则。
- 文档记录工具用来汇编、组织和报告有关系统模型、描述和规格说明以及原型的文档，文档可用于系统所有者、用户、设计人员和构造人员阅读。
- 设计和代码生成器自动生成数据库设计和应用程序或程序的主要部分。
- 测试工具引发事务交互和数据流、度量性能，并提供测试计划和测试脚本的配置管理。

2.4.1.3 正向工程和逆向工程

如前所述，CASE 技术使系统实现自动化建模。如今的 CASE 工具提供了两种不同方式开发系统模型——正向工程和逆向工程。可以把逆向工程看作是从一个现有程序中生成的一张流程图，正向工程

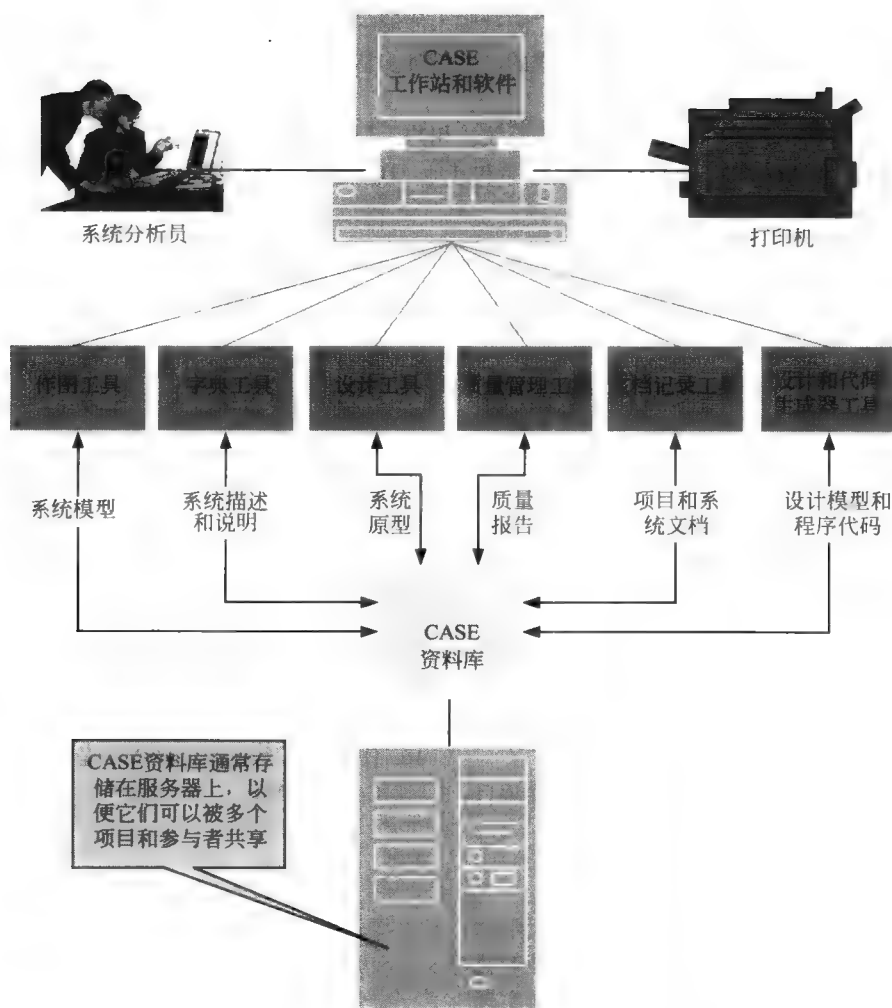


图 2-16 CASE 工具架构

是从流程图直接生成程序。同时将双向工程、正向工程和逆向工程的 CASE 工具称为提供“全程工程”（round-trip engineering）。例如，将一个设计拙劣的系统进行逆向工程，导出一个系统模型，编辑并改进这个模型，然后将改进后的模型进行正向工程，生成一个改进的系统。

2.4.2 应用开发环境

在软件开发过程中对速度和质量的重视产生了 RAD 方法。通过将程序设计语言编译器转换成完整的应用开发环境（ADE），RAD 方法的潜力可以得到放大。ADE 使编程工作变得更简单高效。事实上，多数程序设计语言编译器现如今已集成到 ADE 中。ADE（以及它们支持的编程语言）的例子包括：IBM 公司的 Websphere（Java）、Borland 公司的 J Builder（Java）、Macromedia 的 Cold Fusion、微软公司的 Visual Studio. NET（VB. NET、C#、C++、. NET）、Oracle 公司的 Developer、Sybase 公司的 Powerbuilder。

应用开发环境中包括一些生产率和质量管理的工具。ADE 供应商提供了一些相关工具，第三方供应商则提供了其他许多可以集成到 ADE 中的工具。

- 程序设计语言或解释器是 ADE 的心脏，它们通常提供强大的调试特性和辅助工具，以帮助程序员快速地确定并解决程序设计问题。
- 界面构造工具帮助程序员快速地使用组件库构造用户界面。
- 中间件是一种软件，它帮助程序员集成正在开发的软件及各种数据库和计算机网络。

- 测试工具用于构造和执行测试脚本，测试脚本能够一致且全面地测试软件。
- 版本控制工具帮助由多个程序员构成的团队管理多版本的程序，这既包括开发期间的管理也包括系统实现之后的管理。
- 帮助文件著作工具用于编写联机帮助系统、用户手册和联机培训。
- 资料库链接允许 ADE 同 CASE 工具产品和其他 ADE 与开发工具集成。

2.4.3 过程和项目管理器

第三类自动化工具帮助我们管理系统开发方法学及使用该方法学的项目。CASE 工具和 ADE 用于支持新信息系统和软件的分析、设计和构造活动，项目管理软件和过程管理软件则用于支持跨生命周期活动。微软公司的 Project 和 Niku 公司的 Open Workbench 和 Project Manager 是自动化项目管理工具的例子。过程和项目管理是下一章的内容，在下一章中将介绍更多关于这些自动化工具的知识。

复习题

1. 解释为什么对于企业来说拥有一个标准的系统开发过程很重要。
2. 如何关联系统生命周期和系统开发方法学？
3. 系统开发的 10 个基本原理是什么？
4. 为什么文档在整个开发过程中很重要？
5. 为什么需要过程管理和项目管理？
6. 什么是风险管理？为什么需要它？
7. 大多数项目由哪个关联人员发起？大多数项目的推动力是什么？
8. 在项目定义中，谁是主要的参与者？他们在项目定义中的目标是什么？
9. 在项目定义中三个最重要的发布物是什么？
10. 在需求分析阶段，谁是主要的参与者？为什么他们是主要的参与者？
11. 在决策分析阶段做哪些可行性分析？
12. 什么是模型驱动开发？
13. 为什么流行模型驱动开发？
14. 什么是快速应用开发（RAD）？
15. RAD 可以为系统开发过程带来什么好处？
16. 什么是计算机辅助软件工程（CASE）？举几个 CASE 的例子。

问题和练习

1. 能力成熟度模型（CMM）由卡内基梅隆大学的软件工程研究所开发，广泛应用于私人企业和公共部门。CMM 框架的目标是什么？它如何实现这个目标？
2. 列出 5 个成熟度等级，并做简要描述。
3. 表 2-1 给出了一个典型的项目中工期、人月、质量和费用的差异，这取决于企业的系统开发过程处于 CMM 第 1 级、第 2 级还是第 3 级。按照改进的比率来说，在哪两个等级之间企业获益最大？你认为产生这个现象的原因是什么？
4. 系统开发方法和系统生命周期是两个经常使用但又经常用错的词。这两个词有什么区别？
5. 说明使用系统开发方法如何与 CMM 目标看齐，并能够帮助企业提高成熟度等级。
6. 有几个系统基本原理对所有的系统开发方法都适用。说出这些基本原理，并解释。
7. PIECES 框架由 James Wetherbe 开发，用于作为分类问题的方法。请说出这些分类，然后使用 PIECES 框架对以下问题进行分类：
 - a. 重复的数据存储在整个系统中。
 - b. 需要将一个已有的应用移植到 PDA 设备中。
 - c. 需要自动生成季度销售报告。
 - d. 雇员可以访问人事系统的授权部分。
 - e. 库存系统的用户界面很难使用而且容易产生混淆，导致了高的错误订单率。
8. 项目的每个阶段都包括需要产生并发送到下一个阶段的特定的发布物。使用本书假设的 FAST 方法，需求分析、逻辑设计和物理设计/集成阶段都有哪些发布物？
9. 范围定义是本章所讲方法的第一个阶段，它也是大多数方法学的第一个阶段或第一个阶段的一部分。什么触发了范围定义阶段，哪些关联人员参与这个阶段，哪两个基本问题需要在这个阶段回答，这个阶段需要产生哪三个重要的发布物？
10. 需求分析阶段是系统开发方法学的基本阶段。按照我们的方法，通常哪些关联人员参与这个阶段？需求分析主要关注什么？不关注什么？应该如何评估每个建议的需求？必须避免哪个关键错误？
11. 在我们的方法中，以及大多数系统开发方法学中，系统所有者和系统设计人员不参与需求分析阶段。你认为这样做的原因是什么？

12. 逻辑设计阶段的基本目的是什么？如何实现这个目的？这个阶段涉及技术方案吗？其他方法学中这个阶段常用的同义词有哪些？一般来说谁参与这个阶段？什么是敏捷建模，它有什么作用？这个阶段产生什么发布物？用开发团队的话来说，这个阶段结尾产生什么关键转换？
13. 物理设计阶段的基本目的是什么？谁必须参与这个阶段，谁可以参与？在 continuum 的两端，物理设计的两个基本原理是什么，有什么差别？
- 这个阶段是一个项目可以省掉的阶段吗？可能同其他什么阶段重叠，为什么？
14. 一个顾客要求你的软件开发公司开发一个新的订单处理系统。但是，要求至少发布新系统的最基本部分的时间要求很紧而且不灵活。并且，用户需求很模糊、不清楚。在这个项目中哪两种系统开发策略可能有优势？
15. 采用上一题中描述的开发策略有什么潜在的副作用？

项目和研究

1. 卡内基·梅隆大学的软件工程研究所开发了一系列相关的能力成熟度模型（CMM）。你可以在 SEI 的网站（<http://www.sei.cmu.edu>）读到这些不同的 CMM 产品。
 - a. 说出 SEI 当前维护或开发的 CMM 产品。
 - b. 它们之间有什么不同及相同之处？
 - c. 如果要你使用本书讲述的 CMM 给你的企业分级，或者给你熟悉的企业分级，它处于哪一级？为什么？
 - d. 你建议你的企业采取什么步骤以提高到下一个 CMM 等级？
 - e. 你觉得你的企业的预期收益值得为此提高到一个等级所需的时间、费用和资源吗？为什么值得，或者为什么不值得？
2. 你是一个新的项目经理，并被指派负责一个企业信息项目，该项目涉及你所在企业的每个部门。首席执行官在项目启动会上说成功实现这个项目对于你的企业来说是第一位的。该项目正处于需求分析阶段中期。虽然项目按进度进行，但你注意到系统用户和所有者参加需求会议的安排被取消了。一个更有经验的项目经理告诉你不要担心，这很正常。你应该为此担心吗？
3. 存在许多不同的系统开发方法学在使用，每个都有自己的词汇、阶段数量和范围。在网上查找两到三个有关其他系统开发方法学的信息，然后做如下研究：
 - a. 记录下你找到的系统开发方法学。什么时候、谁、为什么开发它们？
 - b. 它们使用哪些阶段和词汇？
 - c. 绘制一个矩阵将它们的阶段同本书中讲述的 8 阶段方法进行比较。
 - d. 你发现有什么重要差别吗？
 - e. 同本章所讲的 8 阶段方法比较，你觉得你发现的方法学有什么优缺点？
4. 本书中介绍的由 James Wetherbe 开发的 PIECES 框架是一个用于分类问题、机会和指示的框架。联系一个你所在企业、学校或者其他企业的系统分析员。询问他们关于他们企业中使用的信息系统的情况，并描述一般性的问题。选择三个这样的系统：
 - a. 描述你选择的系统，它们存在的问题，以及使用它们的企业。
 - b. 使用书中的 PIECES 框架对这些问题进行分类。
 - c. 描述 PIECES 分类，或者你发现的问题的分类。是否每个问题一般都与一个或多个分类相关？
 - d. 对于不同企业使用的系统，你发现在问题分类上有什么共性？如果你发现分类中的大量共性，你认为这是必然的还是巧合？
 - e. 你认为 PIECES 框架在系统开发生命周期的哪个阶段最有价值？
5. 计算机辅助软件工程（CASE）可以极大地帮助开发人员提高产量、质量，改进文档。对十几个信息技术部门进行一次非正式的调查，调研他们是否使用 CASE 工具，如果使用，了解使用什么类型的工具。另外，调查它们使用 CASE 工具多长时间了，是否在所有项目中使用，还是只在一些项目中使用。可以增加另外一些你觉得有意义的问题。试着将你的调查分成公共部门和私人企业两部分，或者大型企业和小型企业两部分。
 - a. 你调查了哪类企业？
 - b. 你问了什么问题？
 - c. 结果如何？
 - d. 假设调研是受限的、非正式的，你是否能发现任何模式或者趋势？
 - e. 基于你的阅读和调查，你对 CASE 工具的使用有什么看法？
6. 项目经常被取消或放弃，有时是有选择的，有时没有。在网上查找关于项目放弃策略的文章，选

择其中的两篇。

- a. 你选择了什么文章？
- b. 它们关于项目放弃的中心论点、观点和建议是什么？

小型案例

1. 同至少两名项目经理交谈。他们对范围蔓延有什么经验？
2. George 是一个大型公司的 CEO，该公司试图开发一个程序捕捉雇员在计算机上的按键。项目目前已超过预算 100 000 美元，并且落后于进度，而且至少还需要 50 000 美元和 6 个月才能完成。CEO 想继续项目，因为已经投入了那么多。你有什么建议？为什么？
3. Beatrice 是一个出色的经理——她能够管理行政过程并遵循她所在公司的商业规则。她是一个“教条主义者”，总是通过引证说明所做的是“对的”。Beatrice 能成为一个好的项目经理吗？为什么能，或者为什么不能？
4. 公司正试图为库存管理是使用一个现成软件还是开发一个定制软件做决定。现成软件比定制软件便宜，具有所需要的大多数功能。CEO 相信那些缺少的功能可以通过购买软件后 *tweaking* 程序实现。作为该公司的 CIO，你给 CEO 的意见和建议是什么？

团队和个人练习

1. 团队练习：使用不同的通信介质举办一次团队会议。例如：电话、电子邮件、虚拟环境。你注意到技术对会议有什么影响？对于不同的介质，会议的效果一样吗？团队觉得技术如何影响团队关系？
2. 个人练习：分析和设计一个信息系统，完成以后，经常会取消公司中的一些职位。经济理论强烈建议当发生这种情况时应创造新的工作职位，但通常工作的结构化损失和新工作创建之间存在时间差。你对此如何感受？
3. 团队或个人练习：访问一个著名医疗中心的神经紧张治疗部门（例如 UC San Francisco）。写一篇关于涉及的技术和对人的生活的影响的短文。在教师的指导下，在班上分享。

项目管理

本章概述和学习目标

项目管理技能在信息技术界有很大需求。项目管理是对上一章中介绍的系统开发的自然拓展。本章概述应用于系统分析和设计时的一个以过程为中心的主要项目管理工具和技术。本章介绍以下内容：

- 定义术语项目和项目管理，并区别项目管理和过程管理。
- 描述信息系统和技术项目的失败原因。
- 描述项目经理需要的基本能力。
- 描述项目管理的基本功能。
- 区别作为项目管理工具的 PERT 图和甘特（Gantt）图。
- 描述项目管理软件相对于项目管理工具的角色。
- 描述项目管理中的 8 个活动。
- 定义联合项目计划及其在项目管理中的角色。
- 定义项目范围并编写一份工作陈述记录项目范围。
- 使用工作分解结构将一个项目分解成任务。
- 估计任务的持续时间并在 PERT 图上说明任务之间的依赖关系。
- 给一个项目分配资源并使用甘特图生成一份项目进度表。
- 给人员分配任务并指导团队工作。
- 使用关键路径分析调整进度表和资源分配，以响应进度和预算出现的偏差。
- 管理项目的用户期望并调整项目范围。

本章关键术语

项目经理（project manager）是从一个系统项目开始直到结束负责掌管项目的人。成功的项目经理拥有广泛的技术、管理、领导和交流方面的技能。

项目（project）是必须按时在预算内并遵循规格说明完成的一系列活动。

项目管理（project management）是在指定时间内用最少的费用开发可接受的系统的管理过程，内容包括确定范围、计划、人员安排、组织、指导和控制。

过程管理（process management）是记录、管理并持续地改进系统开发过程的活动。

范围蔓延（scope creep）是在信息系统项目进行期间不期望的需求缓慢增加。

特征蔓延（feature creep）指给某个系统不受控制地增加技术特征。

PERT 图（PERT chart）是一种图形化的网络模型，描述一个项目中任务之间的关系。

甘特图（Gantt chart）是一种条形图，以日历为基准描述项目任务。

联合项目计划（Joint Project Planning, JPP）是一种策略，其中项目的所有关联人员参加一个研讨会，就项目决策达成一致意见。

项目范围（project scope）定义项目的边界，项目可能涉及（也可能不涉及）的业务领域。

工作分解结构（Work Breakdown Structure, WBS）是一种图形化工具，用来描述项目分解成开发阶段、开发活动和开发任务的层次结构。

里程碑（milestone）是标识项目主要交付成果的完成的事件。

最优工期（Optimistic Duration, OD）是指估计完成任务所需的最小时间量。

最差工期（Pessimistic Duration, PD）是指估计完成任务所需的最大时间量。

期望工期（expected duration, ED）是指估计完成任务所需的时间量。

最可能工期（Most Likely Duration, D）是指基于最优、最差和期望工期的权重计算完成任务所需的估计时间量。

正向调度（forward scheduling）建立项目开始日期，然后从这个日期开始向前安排进度的项目调度方法。

反向调度（reverse scheduling）建立项目的最后期限，然后从这个日期开始向后安排进度的项目调度方法。

资源调配（resource leveling）是一种改正资源过度分配问题的策略。

关键路径（critical path）是一个相关任务序列，该路径决定了项目最早可能完成的时间。

富余时间（slack time）是一个任务的开始时间和结束时间之间可以忍受的延迟量，这个延迟量不会引起整个项目完成时间上的延误。

变化管理（change management）策略建立一个过程以管理在项目期间出现的变化。

预期管理矩阵（expectations management matrix）是一个理解改变项目参数的动态和影响的工具。

3.1 什么是项目管理

读者中大多数人都熟悉墨菲定律。该定律建议，“任何一件事如果会出错，它就一定会出错。”墨菲定律说明了项目、机器、人和事情为什么会出错。本章将向你讲授信息系统项目的项目管理策略、工具和技术。

信息系统界对**项目经理**的需求一直很强烈。通常，IS 项目经理来自有经验的 IS 开发人员行列，例如系统分析员。虽然你的第一份工作就负责项目管理是不太可能的，但是你应该尽快了解项目管理的过程、工具和技术。最终你将结合这些知识和开发经验，再加上你自己对项目经理的观察，形成你的项目管理职业基础。

在定义项目管理之前，首先应该定义项目。关于项目的定义，有多少作者就会有多少种定义，但我们喜欢由 Wysocki、Beck 和 Crane 给出的定义：

“**项目**是一个（临时的）唯一的、复杂的和关联的具有同一目标或目的并且必须在特定时间里、在预算内、按照规格说明要求完成的活动序列。”^①

以上带下划线的关键词提醒读者注意定义中的要点。当这个定义应用于信息系统开发时，需要注意以下要点：

- 系统开发过程或方法学定义了一个强制的或者可选的活动序列。
- 每个系统开发项目都是唯一的。也就是说，每个系统开发项目都和前一个不同。
- 构成系统开发的活动是相对复杂的。它们需要你应用在本书中学到的技能，还要求你能灵活地运用概念和技能，以适应变化的条件和未预期的事件。
- 到目前为止，你已经知道了构成一个系统开发方法学的阶段和活动一般是按顺序的。虽然一些任务可以重叠，但是许多任务依赖于其他任务的完成。
- 一个信息系统的开发代表了一个目标。为了实现这个目标，开发过程可能需要完成几个任务。
- 虽然许多信息系统开发项目没有绝对的最后期限或者指定的时间期限（有例外情况），但它们经常远远比计划延迟完成。这对于高层管理人员来说一般是不可接受的，所以他们通过组织施加压力来减少产品和业务过程的周期。
- 信息系统很少能在预算内完成。高层管理人员越发抵制这种趋势。
- 信息系统必须满足符合规格说明的企业、用户和管理期望（本书中称之为需求）。

对于任何系统开发项目来说，为了确保项目满足最后期限，在一个可接受的预算内开发，并实现客户的预期和要求，有效的项目管理是必需的。第 2 章介绍了项目管理是一个跨生命周期活动，它覆盖了系统开发方法学的所有阶段。

① Robert K. Wysocki, Robert Beck, Jr., and David B. Crane, *Effective Project Management: How to Plan, Manage, and Deliver Projects on Time and within Budget* (New York: John Wiley & Sons, 1995), p. 38.

良好的项目管理的前提条件是具有一个定义良好的系统开发过程。第2章介绍了用能力成熟度模型（CMM）作为质量管理的框架，该模型的基础是严格的系统开发过程。注意，区分项目管理和过程管理是很重要的。项目管理在前面已经定义过。过程管理是一项不间断的活动，这项活动记录、管理组织所选择的系统开发方法（过程）的使用并改进该方法。过程管理关心应用于所有项目的活动、交付产品和质量标准。过程管理的范围是所有项目，而项目管理的范围是单个项目。本章主要介绍项目管理。

3.1.1 项目失败的原因

什么原因会导致一个项目的成功或者失败？第2章介绍了系统开发的10个基本原理，这些基本原理是所有项目成功的关键因素，请阅读第2章复习这些原理。从项目管理的角度来说，如果一个项目能够达到以下目标，就可以认为是成功的：

- 客户接受最后得到的信息系统。
- “及时”地交付了系统。
- “在预算内”交付了系统。
- 系统开发过程对正在进行的企业运营影响很小。

并非所有的项目都能满足这些标准，仅此，并非所有的项目都是成功的。失败和部分成功项目的数量远远超过了成功的信息系统的数量。项目的不良管理会使得本书讲授的系统分析和设计方法变得毫无意义。我们可以通过研究一些项目经理常犯的错误对项目管理的重要性进行一个正确的评价，下面就让我们看看一些项目的不良管理表现出的问题和后果。

- 没有得到高层管理人员对项目的承诺——有时在项目进行过程中承诺改变了。
- 缺少组织对系统开发方法学的承诺——许多系统开发方法学仅仅是收集了一些“垃圾”。
- 走捷径绕过系统开发方法学——项目团队经常由于下面的一个或多个原因而走捷径：
 - 项目落后于进度表，而项目团队想赶上进度。
 - 项目超出预算，而项目团队想通过省略某些步骤来节省开支。
 - 项目团队没有受过某些方法学的活动和需求方面的培训，或者对此不熟练，所以他们省略了这些步骤。
- 预期管理差——所有用户和管理人员对项目都有预期。随着时间的推移，这些预期可能会发生变化。这会导致两种不期望的情况发生：
 - 范围蔓延是指随着项目的进展，对一个信息系统的用户预期和业务需求不期望地增长。遗憾的是，进度表和预算会受到这种变化的负面影响。
 - 特征蔓延是指一个开发中的系统的技术特征不受控制地增长，而没有考虑到进度和预算。
- 过早承诺固定的预算和进度——在完成详细的问题分析和需求分析之前，你很少能对项目费用和进度做出正确的估计。过早的估计同第2章介绍的逐步投入方法不一致。
- 估计技术差——许多系统分析员使用最佳算法进行估计，然后把得到的结果加倍。这是一种不科学的方法。
- 过于乐观——系统分析员和项目经理往往会很乐观。当项目进度落后时，他们会说，“这不是大问题，我们以后可以赶上。”他们没有认识到某些任务是同其他任务相关的。由于这种关联性，某个阶段或活动中进度的落后会使其他许多阶段和活动相应地落后，这些都会增加费用。
- 人月神话^①——当项目落后于进度时，项目领导常常会试图通过分配更多的人到项目团队中工作来解决问题。这样做不起作用！在时间和人数之间没有线性关系。人员的增加通常会导致更多的沟通问题，使得项目更加落后于进度。
- 人员管理能力不足——管理者往往一头扎进管理工作中，而对管理的责任认识不清。这个问题其实很容易确定，经常的表现是：看上去没有人应该负责；客户不知道项目的状态；项目团队不经常开会讨论和监控进度；项目团队成员之间互相不交流；总是声称项目“完成了95%”。

① Fred Brooks, *The Mythical Man-Month* (Reading, MA: Addison-Wesley, 1975)。

- 没能调整以适应业务的变化——如果在项目进行期间项目的重要性发生了变化，或者如果管理人员或企业发生了重组，就应该重新评估项目与这些变化的兼容性以及项目对企业的重要性。
- 资源不足——这可能是由于错误的估计，或者由于有其他优先的事情要做，或者可能是分配给项目的人员不具备所需的技能或经验。
- 没能“管理计划”——各种因素都可能使得项目经理偏离最初的项目计划。

最后，项目失败的主要原因是大多数项目经理没有接受过作为项目经理的教育或培训。正如优秀的程序员并不总是优秀的系统分析员一样，优秀的系统分析员也并不会自动地成为优秀的项目经理。为了成为一名优秀的项目经理，必须接受“项目管理技术”的教育和技能培训。

3.1.2 项目管理知识体系

项目管理学会是一个指导职业项目经理的培训和认证的专业团体，它发表了项目管理知识体系 (Project Management Body of Knowledge, PMBOK)，用于职业项目经理的教育和认证。本章的内容主要受 PMBOK 的影响。

3.1.2.1 项目经理的能力

优秀的项目经理具有一系列能力，表 3-1 总结了这些能力。这些能力中有些可以在课程上、书本中和专业研讨会上讲授，但是，有些能力则只有从这个领域的专业经验中才能获得。项目管理能力有两个基本前提。首先，你通常无法管理一个你从来没有使用过的过程；其次，如果不理解为该项目提供背景的业务和文化，也无法管理一个项目。

表 3-1 项目经理的能力

能力	解释	如何获得
业务能力		
熟悉业务	将每个系统项目同组织的任务、构想和目标联系起来	业务经验
保持同业务伙伴的关系	在整个系统项目中保持管理人员和用户的参与	业务经验
保证质量	保证每个系统项目对组织整体质量预期做出贡献	业务经验
问题解决能力		
主动性	展示完成工作所需的创造性、计算出的风险和持续性	业务经验
信息收集	巧妙地收集分析、设计和实现信息系统所需的有用信息	第 5 章以及业务经验
分析思维	能够评估和选择合适的系统开发过程，并使用项目管理工具为系统开发做计划、安排进度和做预算	本章
	能够通过分析方法解决问题，将系统分解成它们的组成部分，然后再将各部分组装成改进后的系统	第 7、8、10 章以及业务经验
理论思维	理解系统理论，并将理论应用于信息系统的系统分析和设计中	第 1 章和第 4 ~ 10 章
影响能力		
了解人际关系	理解、认识并响应人与人之间的动机和行为	可以在课程中学习，但需要业务经验
了解组织	理解组织的政策以及如何在一个项目中使用政策	业务经验
预测结果的能力	理解项目决策的含义并管理预期结果和风险	本章中介绍，但需要业务经验
充分利用影响力	巧妙地获得管理人员、用户和技术人员对方案的合作和妥协	业务经验
管理者能力		
激励他人	培训并指导个人克服差异，使大家成为一个团队，共同实现项目目标	业务经验
交流技能	利用会议、报告、备忘录和汇报等各种手段有效地交流，既包括口头交流也包括书面交流	可以在课程中学习，但通常需要业务经验
促进他人发展	确保项目团队的成员受到足够的培训、任命、监管和工作成绩评价，这些工作对完成项目是必需的	业务经验
监督和控制	开发项目计划、进度表和预算，连续地监督进展，并在需要时做出调整	本章中讲授的工具和技术，但需要项目经验

(续)

能力	解 释	如何获得
自我管理能力		
自信	一致地做出决策并坚持决策, 对过程和/或事实具有强烈的自信心	业务经验
管理压力	在压力和反对下有效地工作	业务经验
关心信誉	诚实地做出承诺和方案。在专业领域内保持合适的技术或业务先进性	业务经验
灵活性	能够调整过程、管理风格, 或者根据情况和未预料的问题做出决策	业务经验

资料来源: 摘自 Robert K. Wysocki, Robert Beck, Jr., and David B. Crane, *Effective Project Management: How to Plan, Manage, and Deliver Projects on Time and within Budget* (New York: John Wiley & Sons, 1995)。

3.1.2.2 项目管理职能

管理理论对经理的基本职能已经研究和精炼多年了, 这些职能包括: 确定范围、计划、人员安排、组织、调度、指导、控制和项目总结。

- 确定范围——项目范围定义了项目的边界。项目经理必须确定对项目范围的预期和约束条件, 以便计划活动、估计费用或管理预期结果。
- 计划——计划确定完成项目所需的任务。计划的依据是管理者对项目目标的理解以及对实现目标所使用的方法学的理解。
- 估算——完成项目所需的每个任务必须被估算。将需要多少时间? 将需要多少人? 将需要什么技能? 哪些任务必须在其他任务开始之前完成? 某些任务可以重叠吗? 开销会有多大? 这些都是估算的问题, 其中有些问题可以使用项目建模工具来解决。项目建模工具将在本章后面讨论。
- 调度——给定项目计划, 项目经理负责调度所有的活动。项目进度表应在理解了所需的任务、任务持续时间和任务前提条件的情况下制定。
- 组织——项目团队的成员应该理解他们各自的角色和职责以及他们对项目经理的工作汇报关系。
- 指导——项目一旦开始, 项目经理就要指导团队的活动。每个项目经理都必须展示人员管理能力, 以协调、协商、激励、忠告、赞赏和奖励团队成员。
- 控制——也许控制项目是项目经理最困难也是最重要的职能。计划的执行过程很少能够不出问题或者不发生延误。项目经理必须监视和报告项目的进展, 包括目标、进度和费用, 并且在需要时还要做出合适的调整。
- 项目总结——优秀的项目经理总是在项目结束时总结经验教训。他们从自己的错误中学习, 并对系统开发过程的持续改进制定计划。

所有上述的职能都依赖于项目经理、团队和其他管理人员之间的人际交流。

3.1.2.3 项目管理工具和技术——PERT 图和甘特图

PMBOK 也包括支持项目经理工作的工具和技术, 其中的两个是 PERT 图和甘特图。

PERT (项目评估与评审技术, Project Evaluation and Review Technique) 发展于 20 世纪 50 年代后期, 用于为美国海军计划和控制大型的武器的开发项目。PERT 图是一种图形化的网络模型, 描述一个项目中任务之间的关系。图 3-1 展示了一个 PERT 图的例子。PERT 图用来在任务被调度之前弄清项目任务之间的依赖关系。图中的方框代表项目任务 (在图中我们使用了第 2 章的项目开发阶段), 可以通过调整方框中的内容反映各种项目属性, 例如进度和实际的开始和结束时间。箭头指示了一个任务依赖于另一个任务的开始或完成。

甘特图最早由 Henry L. Gantt 在 1917 年提出, 是最常用的项目调度和进展评估工具。甘特图是一种简单的水平条形图, 它以一个日历为基准描述项目任务。每个条形表示一个命名的项目任务, 任务名称垂直地列在左边的列中, 水平轴是日历时间线。图 3-2 显示了一个项目开发阶段级的甘特图, 内容也是基于第 2 章的项目开发阶段。注意, 我们使用了图 3-1 中显示的同一个项目。

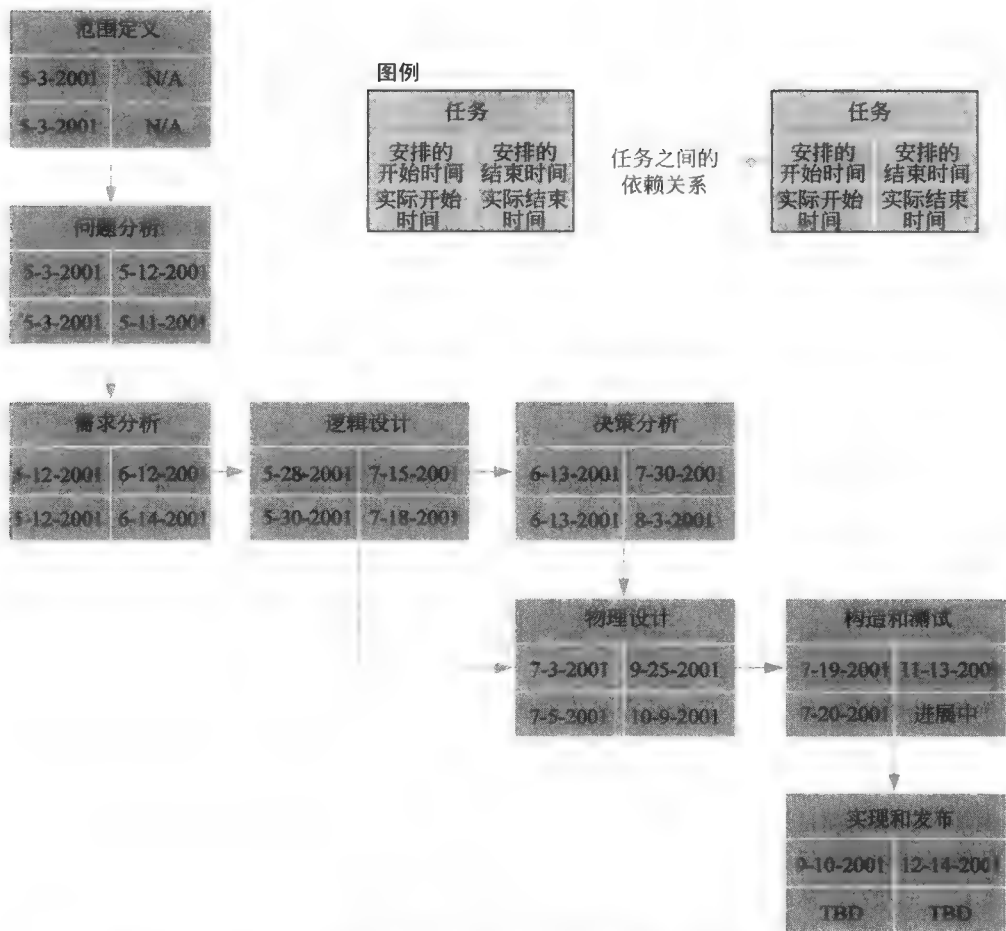


图 3-1 一张 PERT 图

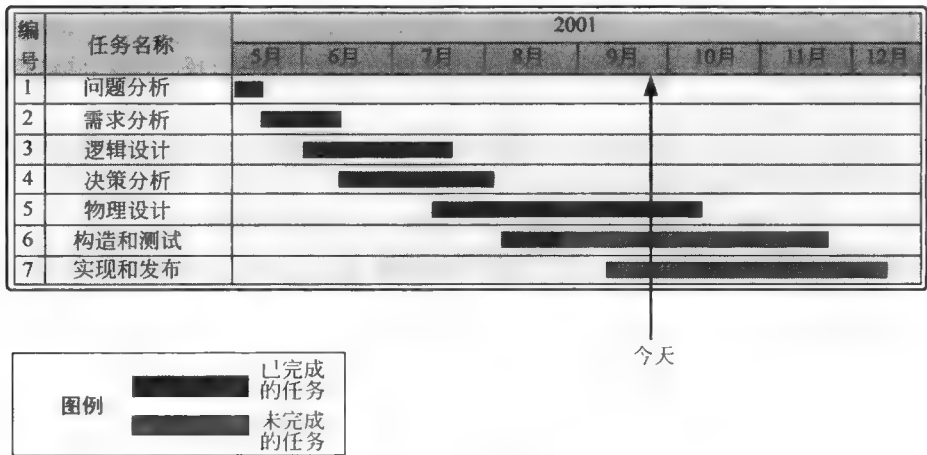


图 3-2 一张甘特图

甘特图的优点是可以清楚地显示重叠任务，即可以同时执行的任务。图中的条形可以加阴影，以清楚地指示任务完成的百分比和项目进展情况。从图中一眼就可以看出哪个阶段提前于进度或者滞后于进度。甘特图的流行是由于它的简洁性——容易学习、阅读、制作和使用。

甘特图和 PERT 图并非是互斥的。当你交流进度时甘特图更有效，当你想研究任务之间的关系时 PERT 图更有效。

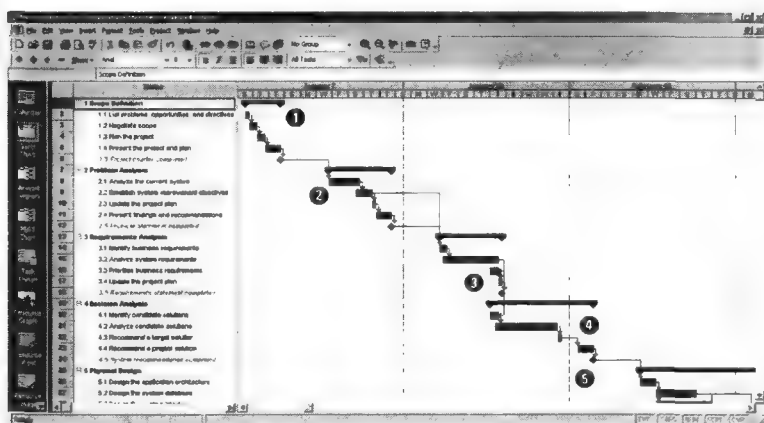
3.1.2.4 项目管理软件

项目管理软件用来帮助项目经理计划项目，制定进度表，制定预算，监视进展和费用，生成报告和做出必要的修改。有代表性的自动化项目管理软件包括：Niku 公司的 Project Manager、Artemis International Solutions Corporation 公司的 9000、Computer Associate 公司的 AIFusion Process Management Suite、微软公司的 Project、Primavera 公司的 Project Planner 和 Project Manager、C/S Solutions 公司的 Risk +。

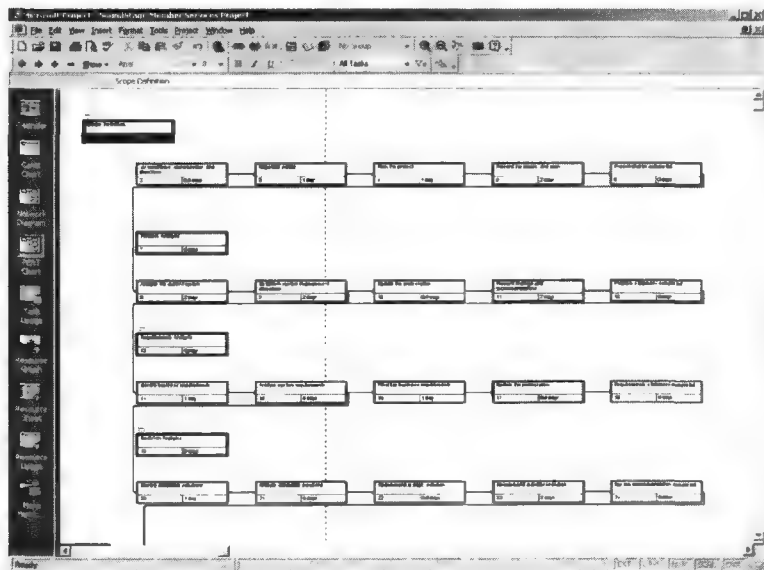
我们将使用项目管理软件讲授项目建模和管理技术。我们之所以使用 Project 是因为这个工具可以通过大学书店按照特殊的价格提供给学生和研究所使用。Project 像大部分项目管理软件工具一样，既支持 PERT 图也支持甘特图。

图 3-3a 展示了音阶公司会员服务项目的一张 Microsoft Project 甘特图。请注意以下数字注解：

- ①该线条表示总成型任务（summary tasks），代表项目开发阶段，它将被进一步分解成其他任务。
- ②该线条指示任务是进度中的“关键”任务，意味着这些任务的任何延误都将会延误其他任务和整个项目进度。后面将更详细地讨论关键任务。
- ③该线条指示任务不是进度中的关键任务，意味着它们有一些富余时间，这些任务的延误将不会影响其他任务和整个项目进度。



a)



b)

图 3-3 Microsoft Project 甘特图和 PERT 图

- ① 这个箭头指示两个关键任务之间的前提条件（另一个箭头指示两个非关键任务之间的前提条件）。
- ② 这个菱形表示里程碑——没有持续时间的事件。它们表示某些重要任务或交付产品的结束。

图 3-3b 展示了一张根据甘特图中举例的那个项目计划做出的 Project PERT 图。任务矩形中每个单元的内容可以在 Microsoft Project 中都随意定制。

3.2 项目管理生命周期

第 2 章中介绍的能力成熟度模型（CMM）定义了一个用来评估组织的信息系统开发活动的质量框架。CMM 第 1 级定义为“初始级”，特点是没有一致的项目或过程管理功能。改进成熟度的第一个阶段就是实现一致的项目管理功能——称为 CMM 第 2 级。在本节中，我们介绍一个符合 CMM 第 2 级成熟度的项目管理生命周期。

图 3-4 显示了一个项目管理过程或生命周期。项目管理是一个跨生命周期活动，也就是说，项目管理活动同第 2 章中介绍的所有系统开发阶段都有重叠。图 3-4 中展示的项目管理活动分别对应典型的管理职能：确定范围、计划、估算、调度、组织、指导、控制和项目总结。

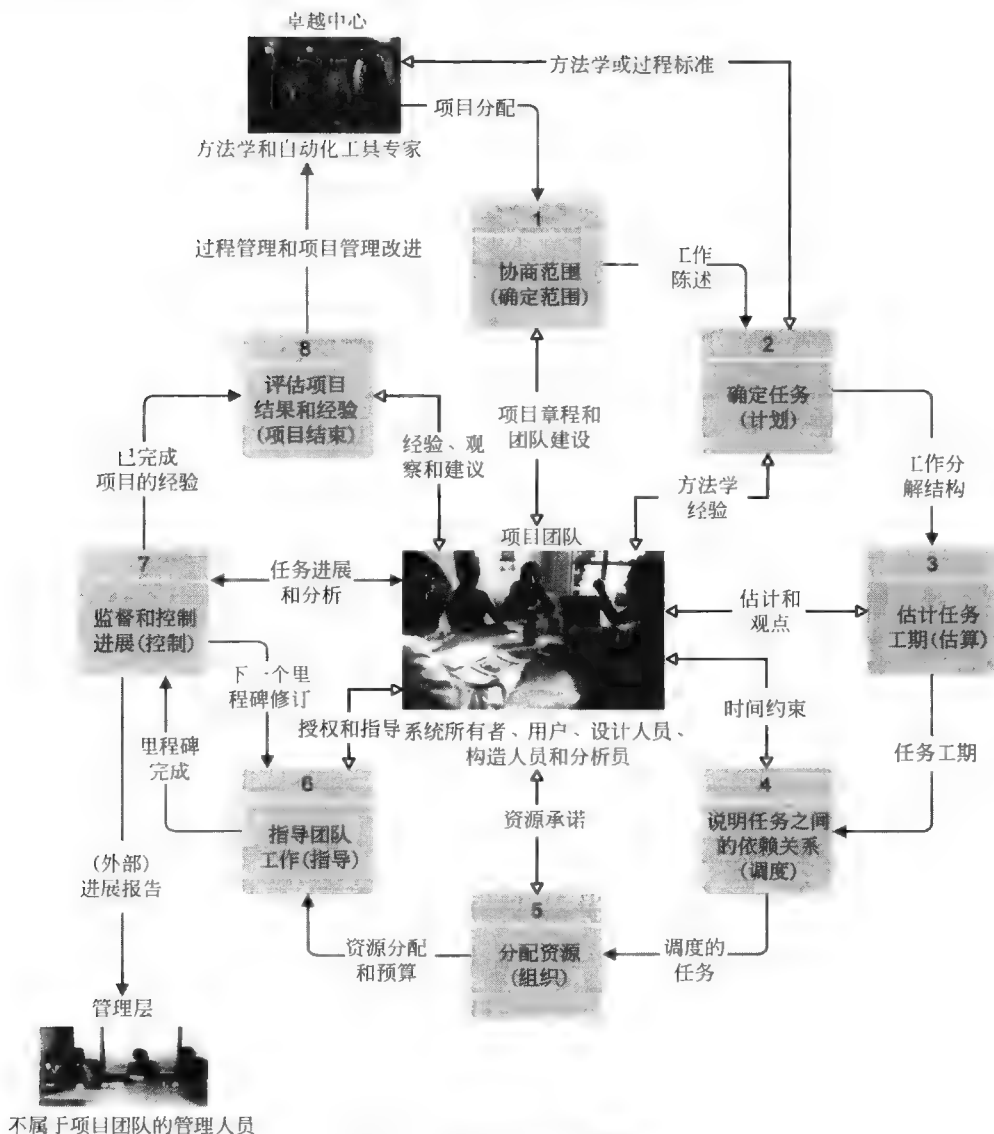


图 3-4 项目管理生命周期

图 3-4 显示的项目管理过程包含联合项目计划技术^①。联合项目计划 (Joint Project Planning, JPP) 是一种策略, 其中项目的所有关联人员 (包括系统所有者、用户、分析员、设计人员和构造人员) 参加一个 1~3 天的项目管理研讨会, 会议的目的是就项目范围、进度表、资源和预算达成一致意见 (可能还需要后续的研讨会或会议来调整范围、预算和进度)。注意, 在 JPP 中, 项目团队主动地参与到所有项目管理活动中。

在下面的各小节中, 我们将概述图 3-4 中列举的每个项目管理活动, 并讲授如何使用合适的项目管理工具和技术。

3.2.1 活动 1——协商范围

对于有效的项目管理来说, 也许最重要的前提条件出现在项目的开始阶段。在确定和安排任务或为任务分配资源 (人力) 之前, 所有有关各方必须就项目范围达成一致意见。项目范围定义了一个项目的预期, 项目预期最终决定对项目成功的满意度。相应地, 项目范围的协商是项目管理生命周期中一个必需的活动。项目范围是什么? 项目范围定义项目的边界——哪些业务将被研究、分析、设计、构造、实现并最终得到改进。项目范围也定义了一个系统的哪些方面被认为是在项目以外。对以下 5 个基本问题的回答将影响到项目范围的协商:

- 产品——你想要什么?
- 质量——你希望它有多好?
- 时间——你想什么时候得到它?
- 费用——你愿意投入多少资金?
- 资源——你愿意或者能够拿出多少资源?

对上述要素的协商是一个妥协和交换的过程, 将包括许多个回合。这项活动的结果是获得一致同意的工作陈述, 描述项目中要完成的工作。在咨询合约中, 工作陈述已成为咨询顾问和客户之间一种常用的合同。这个方法对于内部系统开发项目同样有效, 可以用来建立业务管理人员和项目经理及团队之间的约定。

图 3-5 显示了一份典型的工作陈述文档的提纲。文档的长短将因组织而异, 它既可以短到一两页, 也可以长达好几页。

工 作 陈 述	
I. 目的	C. 培训需求
II. 背景	D. 会议进度
A. 问题、机会或指示陈述	E. 汇报方法和频率
B. 导致项目需求的历史	F. 冲突管理
C. 项目目标和目的	G. 范围管理
D. 产品描述	VI. 约束条件
III. 范围	A. 启动日期
(注意信息系统构件的使用)	B. 最后期限
A. 关联人员	C. 预算
B. 知识	D. 技术
C. 过程	VII. 大致估计
D. 通信	A. 进度表
IV. 项目方法	B. 预算
A. 开发路线	VIII. 满意条件
B. 交付成果	A. 成功的准则
V. 管理方法	B. 假设
A. 团队组建的考虑	C. 风险
B. 管理者和经验	IX. 附录

图 3-5 一份工作陈述提纲

① Wysocki, Beck, and Crane, *Effective Project Management: How to Plan, Manage, and Deliver Projects on Time and within Budget*, p. 38.

3.2.2 活动2——确定任务

给定了项目范围，下一个活动就是确定项目任务。这个任务要确定需要做的工作，通常，这种工作以自顶向下的纲要方式定义。在第2章中，我们学习了系统开发路线和它们的各个开发阶段。但开发阶段对于计划和调度一个项目来说太大也太复杂，我们需要将开发阶段分解成开发活动和开发任务，直到每个开发任务表示了一个可以进行计划、调度和分配的可管理的工作量为止。某些专家提倡分解任务直至开发任务表示了可以在两个星期或更短时间内完成的工作量为止。

项目经理将最终决定纲要中的详细程度。但是，大多数系统开发方法学为你分解了开发阶段——分解成建议的开发活动和开发任务。这些开发活动和开发任务不必一成不变，也就是说，大多数方法学允许根据每个项目的独特性对开发活动和开发任务进行一定的增加、删除和修改。用于确定和记录项目开发活动和开发任务的一种流行工具是工作分解结构。工作分解结构（Work Breakdown Structure, WBS）是指将项目层次化地分解成开发阶段、开发活动和开发任务。

工作分解结构可以绘制成类似于组织结构图的自顶向下的层次图（见图3-6）。但在Project中，WBS使用简单的纲要风格描述，即项目的甘特图“视图”中的开发活动和开发任务的缩排。Project也提供一种标准编号方案，用来表示项目的层次分解，如下所示：

- 1 项目阶段 1
 - 1.1 阶段 1 中的活动 1
 - 1.1.1 阶段 1 中活动 1 的任务 1
 - 1.1.2 阶段 1 中活动 1 的任务 2
 - 1.2 阶段 1 中的活动 2……
 - 2 项目阶段 2……

如果回顾一下图3-3a，将会注意到Project在甘特图中为WBS提供了一系列显示。请注意缩排和编号的使用，它们区分了任务和子任务。

我们可能希望在WBS中包含特殊的称为里程碑的任务。里程碑是标识项目开发期间主要交付成果的完成或结束的事件。在信息系统项目中，里程碑的例子有：与产生一个主要交付成果（如需求陈述，见第2章）相关的所有开发任务的结束。在WBS中使用专门的格式（例如使用斜体）区分里程碑和其他任务也许是有用的。

3.2.3 活动3——估计任务工期

给定一个详细程度合适的工作分解结构，项目经理必须估计每个开发任务的工期。任务工期是一个随机变量，其值取决于一些因素，例如：团队规模、用户数量、用户可用性、用户态度、业务系统的复杂性、信息技术架构、团队人员的经验、对其他项目投入的时间以及其他项目的经验。

大部分系统开发方法学不仅定义开发任务，而且提供任务工期的基线估计。项目经理必须把这些基线估计调整成对每个特定项目合理的实际估计。

在Microsoft Project中，一套方法学的所有开发阶段、开发活动和开发任务都简单地称为任务。工作分解结构则既包括总成型任务，也包括基本任务。总成型任务是包含其他任务（例如开发阶段和开发活动）的任务。基本任务是不包含任何其他任务的任務。项目经理必须估算工期的任务就是这些基本任务（像大多数项目管理软件一样，Microsoft Project将自动地根据基本任务的估计工期计算所有总成型任务的工期）。

对于那些非里程碑的基本任务，我们必须估计任务工期。当估计任务工期时，理解实耗时间的概

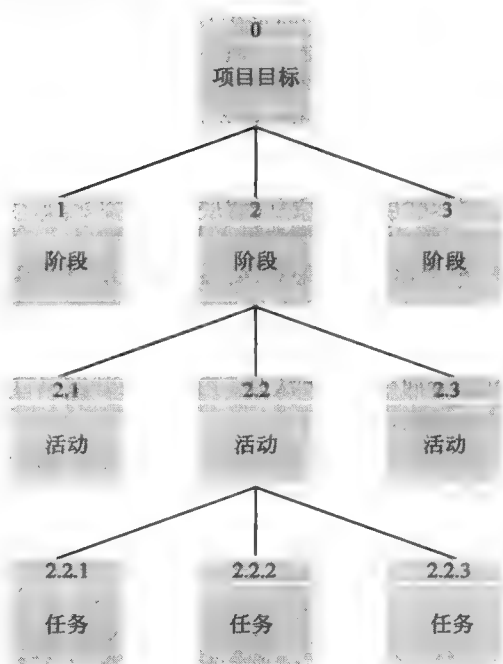


图3-6 一个图形化的工作分解结构

念很重要。实耗时间考虑了两个重要的人员因素：

- 效率——没有人能够以百分之百的效率工作。大多数人都要短暂休息、午休、上厕所，还要花时间阅读他们的电子邮件、检查他们的日程安排、参与非项目开发的工作，甚至是闲聊。专家和普通员工的差别在于工作效率，但一个常用的值是 75%。
- 中断——人们都经历过接电话、接待来访者和其他计划外事情的打断，这增加了项目开发工作所需的实际时间。这对于不同的员工各不相同，中断可能消耗掉你每天时间的不到 10%，也可能大于 50%。

这个概念为什么重要？假定一个任务以百分之百的效率和在没有中断的情况下可以在 10 小时内完成，并且假定一个工作人员的效率是 75% 和 15% 的中断，则任务真正的估计时间将是：

$$10 \text{ 小时} \div 0.75 \text{ 效率} = 13.3 \text{ 小时} \div (1.00 - 0.15 \text{ 中断}) = 15.7 \text{ 小时}$$

有许多技术可以用来估计任务工期。为了演示说明的目的，我们介绍以下传统技术：

1. 估计完成任务所需的最小时间量。我们称之为**最优工期 (OD)**。最优工期假定甚至最可能发生的中断和延误（例如，偶然的雇员生病）也不会发生。
2. 估计完成任务所需的最大时间量。我们称之为**最差工期 (PD)**。最差工期假定几乎任何会出错的事都会出错。所有可能的中断或延误的假定都是不可避免的，例如劳工罢工、生病、培训、不正确的需求说明、设备发货延迟以及过低估计系统复杂性。
3. 估计完成任务所需的**期望工期 (ED)**。不要只是取最优工期和最差工期的中间值，应该确定最可能发生的中断和延误，例如偶然的雇员生病、无经验的人员和临时的培训。
4. 计算**最可能工期 (D)**，计算公式如下：

$$D = \frac{(1 \times OD) + (4 \times ED) + (1 \times PD)}{6}$$

其中，1、4 和 1 是用来计算三个估计值的加权平均值的默认权重。

计算 OD、PD 和 ED 可能是需要技巧的，而且要有经验。有几种技术常用于估计，以下是三种最常用的技术：

- 分解——这是一种简单的技术，其中一个项目被分解成小的、可管理的任务块，每个任务块可以根据过去项目和类似复杂程度的任务块的历史数据进行估计。
- COCOMO——这是一种基于模型的技术，它根据以前项目的标准参数估计新项目及其任务的工期。
- 功能点——这是另一种基于模型的技术，其中一个项目的“最终产品”根据输入、输出、文件和查询的数量和复杂程度进行估计。功能点的数量则与具有类似功能点数量的项目进行比较，以估计工期。

一些自动化项目管理工具（例如 CS/10000 和 Cost · Xpert）提供了专家系统技术，可以根据你对特定问题的回答做出估计。

里程碑（定义见上一小节）没有工期，它们只是发生。在 Microsoft Project 中，通过设置工期为零来表示里程碑（在甘特图中，这些零工期任务的图形符号从线条变成了菱形）。

3.2.4 活动 4——说明任务之间的依赖关系

给定所有任务的工期估计之后，我们可以开始制定一个项目进度表。项目进度表不仅取决于任务工期，而且取决于任务之间的依赖关系。换句话说，某个任务的开始和结束可能依赖于其他任务的开始和结束。共有 4 类任务之间的依赖关系：

- 完成到开始 (FS) ——某个任务的完成触发另一个任务的开始。
- 开始到开始 (SS) ——某个任务的开始触发另一个任务的开始。
- 完成到完成 (FF) ——两个任务必须同时完成。
- 开始到完成 (SF) ——某个任务的开始标志另一个任务的完成。

任务之间的依赖关系既可以通过甘特图也可以通过 PERT 图建立和描述。图 3-7 描述如何在 Project

的甘特图视图中输入任务之间的依赖关系。请注意以下数字注解。

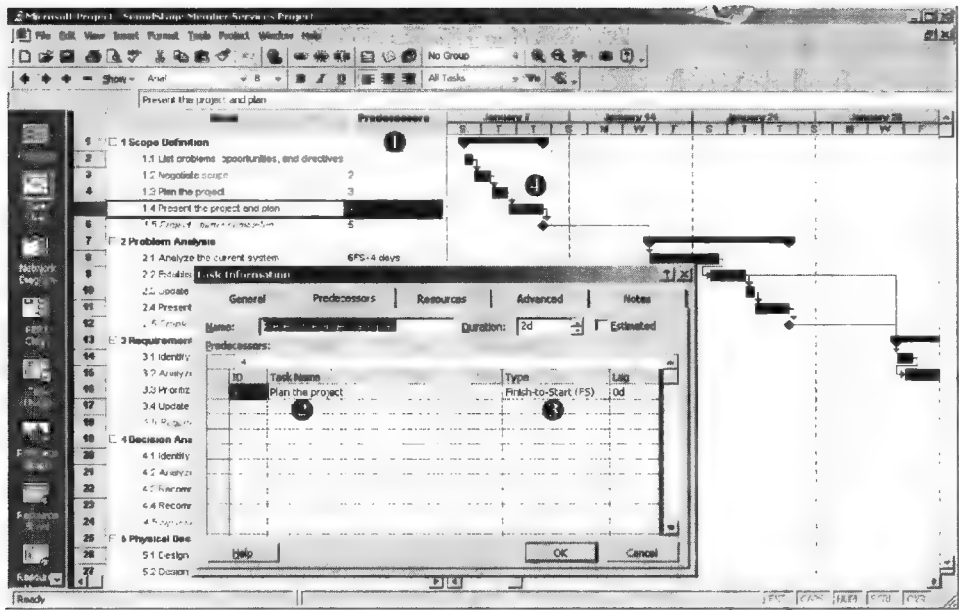


图 3-7 在 Project 中输入任务之间的依赖关系

- ❶在甘特图视图中，任务之间的依赖关系的输入是将依赖的任务行号输入到 Predecessors（前导）列中。注意一个任务可以有零个、一个或多个前导任务。
- ❷任务之间的依赖关系也可以通过打开一个给定任务的 Task Information 对话框输入（或修改）。
- ❸任何给定的相关任务之间的依赖关系类型都可以在 Task Information 对话框中输入。
- ❹任务之间的依赖关系在甘特图中图形化地表示为代表每个任务的条形之间的箭头。箭头可以在左边（指示“开始”依赖关系），也可以在右边（指示“完成”依赖关系）开始或结束。

里程碑（如前面的定义）几乎总是有几个前导任务，这些任务必须在你宣布里程碑已经实现之前完成。

给定一个项目的开始时间、要完成的任务、任务工期和任务之间的依赖关系后，现在就可以调度项目。有两种方法用于调度安排：

- 正向调度建立项目开始日期，然后从这个日期开始向前安排进度。根据任务而定的计划工期、它们的相互关系以及为完成这些任务分配的资源，计算一个计划的项目完成日期。
- 反向调度建立项目的最后期限，然后从这个日期开始向后安排进度。任务、工期、相互关系和资源必须被考虑，以确保项目可以按照最终期限完成。

每个任务都可以被指定开始日期和结束日期。像大多数项目管理工具一样，当你输入了任务工期和任务之间的依赖关系（前导任务）时，Project 实际上为你安排了进度表。在甘特图中，任务条被扩展以反映工期，并左右移动以反映开始日期和结束日期。Project 也可以生成最终进度表的一个传统日历视图，如图 3-8 所示。

3.2.5 活动 5——分配资源

上一步得到了“一张”进度表，但这还不是“最终的”进度表。我们还需要考虑对项目的资源分配。资源包括以下内容：

- 人——包括所有将以任何形式参与该项目的系统所有者、用户、分析员、设计人员、构造人员、外部代理和办事员。
- 服务——某种会按照每次使用基础收费的服务，例如质量检查。
- 工具和设备——包括完成项目需要的所有房间和技术。

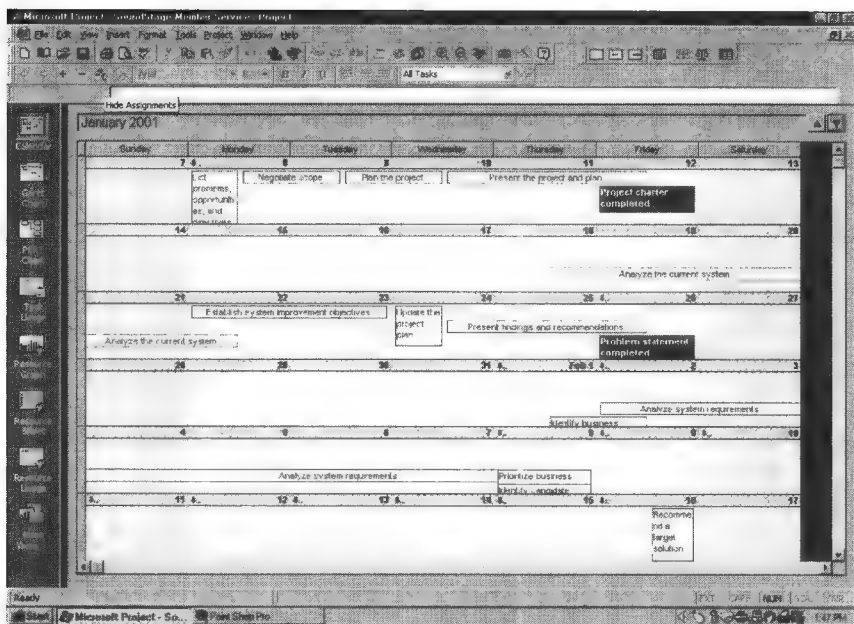


图 3-8 日历视图的项目进度表

- 供应和材料——铅笔、纸张、笔记本、灯座等多样东西。
- 经费——所有上述内容都转换成预算的资金。

资源（特别是人力和设备）的可用性可以极大地改变项目的进度。

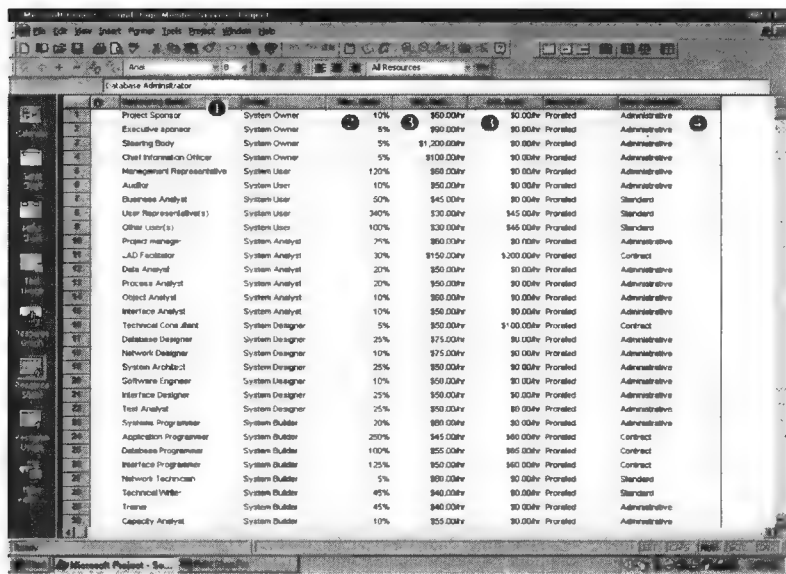
大多数系统开发方法学按照角色确定每个任务所需的人力资源。一个角色不同于一个职务头衔。可以将角色看作是一顶“帽子”，某些人戴它，是因为他们具有相应的技能。任何一个人都有可能戴许多顶帽子（这样就扮演了多个角色）。而且，许多人可能拥有扮演一个特定角色所需的技能。项目经理的任务就是分配特定的人充当特定的角色，或者从管理层获得承诺提供人员来充当特定的角色。FAST方法学中有代表性的角色包括：审核员、业务分析员、业务主题事务专家、数据库管理员、主要负责人、信息系统经理、JAD 主持人、JAD 书记员、管理部负责人、网络管理员、程序员、项目经理、系统建模员。

在 Project 中，角色和任务分别在 Resource Sheet 视图中指定，如图 3-9a 所示。在所选择的方法学和开发路线模板中可能有预定义的角色和资源可用。

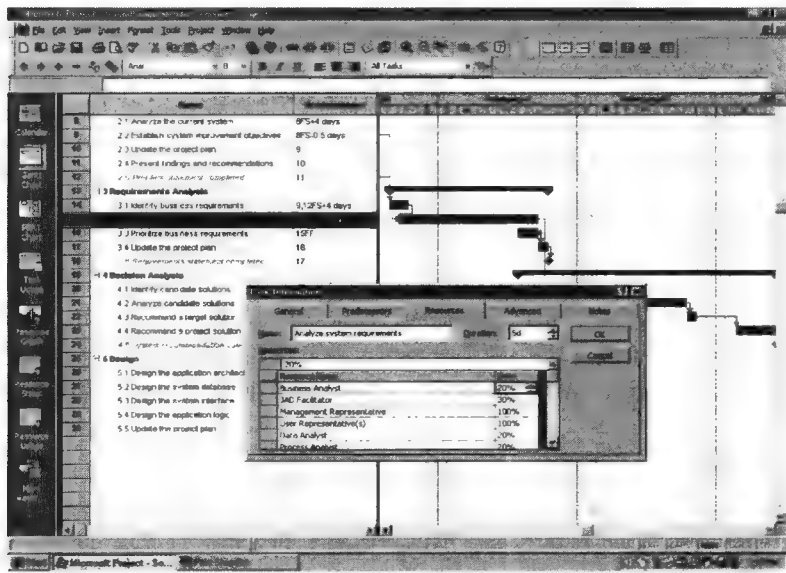
- ① 项目经理在 Resource Name 列输入人[角色]的名字和头衔。资源还可以包括特定的服务、工具、设备、供应、材料等。
- ② 注意，Resource Sheet 提供了一列，用于设置将一个资源的百分之多少分配给该项目。例如，一个数据库管理员可以安排 $\frac{1}{4}$ （25%）的时间用于该项目。大于 100% 的分配说明需要多个人来充当项目中一个给定的角色。例如，通过为资源设置 Max. Units 为 250%，你将说明需要相当于 $2\frac{1}{2}$ 的全职程序员。
- ③ Project 也允许项目经理估计每个资源的费用。这些费用可以根据公司历史数据、咨询合同或内部的费用计费标准进行估算。注意，标准费用和超支费用都可以估算出来。为了保护实际工资信息，这些费用通常根据标准估算。
- ④ 每个资源都有一个 Calendar 列，它考虑了标准的工作周和节日以及个人的假期和其他任务。

现在我们可以把资源具体地分配到任务，如图 3-9b 所示。当资源分配到任务时，项目经理将说明为完成每个分配任务所需资源的单位（这可能是那个任务所需的一个人的工作时间的百分比）。

当正式分配这些资源之后，将调整进度表（在 Project 这样的工具中是自动进行的）。如果你输入了资源的费用，Project 工具也将自动地根据资源和进度计算并维护一个预算。



a)



b)

图 3-9 定义和分配项目资源

3.2.5.1 给任务分配人员

补充“合适的”团队成员可能成就一个项目，也可能毁掉一个项目。以下是针对选择和补充团队成员的指南：

- 补充有天赋的、积极性高的人。能力强且积极的团队成员更能够在没有帮助的情况下克服项目的障碍，也更有可能满足项目的最后期限并保证工作质量。
- 为每个人选择最合适任务。所有的工作人员都有强项和弱项，能干的项目经理会挖掘团队成员的力量，同时避免给团队成员分配他不具备相应技能的任务。
- 促进团队融洽。选择将会互相补充并合作融洽的团队成员。
- 为未来做计划。引入有潜力被项目领导指导的新手。新手可能不如老手能干，但你需要他们，并且在未来的项目中必须仰仗他们。

- 保持小规模团队。通过限制团队规模，将减少沟通负担和难度。一个两人的团队只有1条沟通路径，一个4人的团队有6条沟通路径，一个50人的团队至少有1200条沟通路径。沟通路径越多，沟通问题增加的可能性越大。根据同样的理由，团队应该足够大，以便如果失去一位团队成员，在关键技能方面仍能够提供足够的备份和支持。

3.2.5.2 资源调配

到目前为止，我们已经确定了任务、任务工期和任务之间的依赖关系，并且给每个任务分配资源，以制定项目进度表。当给任务分配资源时，过度分配资源很常见。所谓过度分配，是指在项目的某个时间，我们分配了超过我们可以提供的资源。

例如，在项目的一个特定时期（天、星期等），我们可能分配一个人完成多项任务，加起来超过了这个人在那个时期可以工作的时间。这说明整个进度不可行，因为过度分配的资源不可能合理地按照进度完成所有分配的任务。为了改正这个错误，项目经理必须使用一种称为资源调配的技术。资源调配是一种策略，用于通过延迟或分解任务改正资源过度分配问题。下面简单解释一下这两种方法。

延迟任务是基于关键路径和富余时间的概念。当考虑项目进度时，某些任务比其他任务对进度延误更敏感。因此，项目经理必须了解项目的关键路径。项目的关键路径是一个相关任务序列，该序列具有最大总和的最可能工期。关键路径决定了项目最早可能完成的时间（我们前面描述了如何估算一个任务的最可能工期）。这些关键路径任务没有可用的富余时间。因此，关键路径上任何任务完成时间的延误都将引起整个项目完成时间上的整体延误。同关键任务相对立的是一些有富余时间的任务。任何非关键任务可用的富余时间是一个任务的开始时间和结束时间之间可以忍受的延迟量，这个延迟量不会引起整个项目完成时间上的延误。具有富余时间的任务滞后于进度的时间量只要小于或等于富余时间，就不会对项目的最后完成日期有影响。在某些任务中，富余时间的存在为我们提供了延迟任务开始时间的机会，可以调配资源而不影响项目的完成时间。当然，除非可以分解任务，才可能延迟一个关键路径的任务来调配资源。

分解任务包括将一个任务分解成多个任务，并为这些任务分配不同的资源。这样，一个资源过度分配的任务现在可以指定（假定）没有过度分配的两个或多个资源。分解任务需要确定和分配新的资源，例如分析员、联络员或咨询顾问。

手工地进行资源调配是很枯燥乏味的。项目经理需要知道每个资源可用于项目的总时间、该资源所有的任务分配和各个时间段中这些任务分配的总工期。所有的项目管理软件工具（如 Project）都可以自动地确定关键路径和富余时间。这使得同样的软件工具可以跟踪资源分配并自动地进行资源调配。如今的项目经理很少手工调配资源。

资源调配将是一个持续不断的活动，因为进度和资源分配可能在一个项目开发过程中不断变化。

3.2.5.3 进度表和预算

给定基于调配资源之后的进度表以及每个资源的费用（例如，一个系统分析员或数据库管理员每小时的费用），项目经理就可以生成一份打印的（或者基于 Web 的）文档，用来同所有有关各方交流项目计划。项目管理工具将提供一个项目的多个视图，例如日历图、甘特图、PERT 图、资源和资源调配报告、预算报告。所有剩下的工作就是指导资源完成项目任务。

3.2.5.4 交流

工作陈述、主要交付成果的时间表和整个项目进度表将同所有对项目感兴趣的各方交流。该交流还应该包括一个用于汇报进展的计划（既包括口头汇报也包括书面汇报）、交流的频率以及提交反馈和建议的各方的联系人和联系方式。公司的内联网是使每个人保持了解项目进展和问题的一种有效方式。

3.2.6 活动6——指导团队工作

所有前面的项目管理活动生成项目的一个主计划，现在是执行计划的时候了。指导团队工作包括几个方面的内容。Tom Demarco 在他的《The Deadline: A Novel about Project Management》一书中说到，管理中最困难的工作是人员管理。

新项目经理很少是熟练的人员管理者。大多数人从作为下级的经验中学习管理——那些关于管理

他们的人的事（他们喜欢的和不喜欢的）。这个主题需要一章的篇幅。在下面的列表中，我们提供了一个典型的项目管理建议列表，该列表源自 Keith London 的书《The People Side of Systems》。

给项目领导的十条提示：	
保持一致	设置现实的最后期限
提供支持	设定可见的目标
不要给出你做不到的承诺	解释并说明，而不仅仅是做
公开表扬，私下批评	不要依赖差不多的 [状态报告]
掌握士气	鼓励良好的团队精神

正如 McLeod 和 Smith 提到的：“来到一个系统开发团队中的人并不会立即形成一个紧密的单位。”他们认为一个项目团队将走过团队开发的各个阶段，如图 3-10 所示。

对于任何一位第一次管理他人的经理，Kenneth Blanchard 和 Spencer Johnson 的《The One Minute Manager》都是经典的、有趣的和不可缺少的。书中介绍了管理人员通过下属的工作实现成功的秘诀。

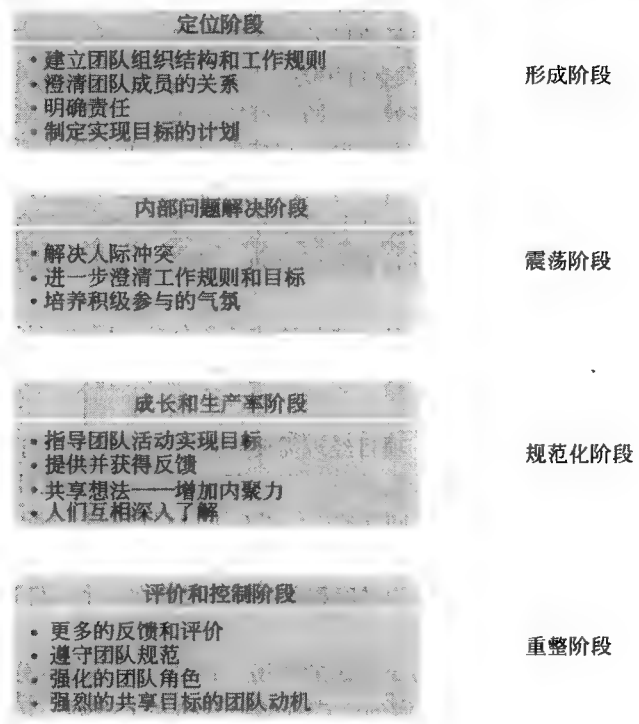


图 3-10 团队成熟的阶段

资料来源：摘自 Graham McLeod and Derek Smith, *Managing Information Technology Projects* (Cambridge, MA: Course Technology, 1996)。

3.2.7 活动7——监督和控制进展

当执行项目时，项目经理必须控制项目，也就是说，监督项目的进展是否符合范围、进度和预算。管理者必须汇报进展情况，当需要时调整范围、进度和资源。

3.2.7.1 汇报进展

应该频繁汇报进展以便于管理，但又不能太频繁而成了项目开发工作的一种负担和阻碍。例如，Keane 公司建议进展汇报或会议每两周举行一次——同公司的项目计划策略保持一致，该策略将项目分解成所需工作时间不超过 80 个小时的任务。

项目进展汇报可以是口头的或者书面的。图 3-11 演示了一个书面项目进展报告的模板。项目进展报告（或演示报告）应真实和正确，哪怕消息并不太好。项目进展报告应该汇报成功，但也应该清楚地确定问题和担心，这样才能在它们发展成大问题或灾难之前将其解决。

项目进展报告	
I. 封面	IV. 以前的问题和结果
A. 项目名称或标识	A. 采取的行动和问题现状
B. 项目经理	B. 新的或改进的行动
C. 汇报日期	1. 建议
II. 进展总结	2. 责任分配
A. 进度分析	3. 最后期限
B. 预算分析	V. 新的问题和结果
C. 范围分析	A. 问题
(描述任何可能对将来进展有影响的项目范围变化)	(实际的或预期的)
D. 过程分析	B. 结果
(描述策略和方法学遇到的任何问题)	(实际的或预期的)
E. 项目进展甘特图	C. 可能的解决办法
III. 活动分析	1. 建议
A. 从上次汇报以来完成的任务	2. 责任分配
B. 当前任务和交付成果	3. 最后期限
C. 短期内的任务和交付成果	VI. 附件
	(包括相应的来自项目管理软件的打印输出材料)

图 3-11 一份进展报告的提纲

当任务完成时，项目进展可以记录在 Project 中（见图 3-12）。请注意甘特图中的数字注解：

- ① 如覆盖每个任务条全长的白线所指示的，范围定义阶段的所有任务都已经完成了。注意因为这些任务已经完成，所以它们就不再是关键任务——相应的任务条从浅色变成深色。
- ② 在问题分析阶段，只有第一个任务（Analyze the current system）百分之百地完成了。
- ③ 注意“Establish system improvement objectives”任务条中有一部分是白线，长度为任务条总长度的 60%，这说明该任务完成了大约 60%。任务条仍然是浅色的，因为该任务的任何延迟仍将影响项目的完成日期。
- ④ 图表中所有剩下的任务还没有开始。当任务处于开始、进行中或完成时，实际的进展都会被记录下来。
- ⑤ 任何给定任务的进展都记录在该任务的 Task Information（任务信息）对话框中。在本例中，项目经理正在记录已命名任务完成了 10%。

Project 也提供了一些可定制的预定义报告，这些报告可以用来表示有用的项目状态信息。

3.2.7.2 变化管理

即使在计划过程的早期已经就一个正确完成的工作陈述达成了一致，范围增长失去控制的现象仍很常见。范围增长称为“变化”。例如，Keane 公司注意到：“变化常常是客户和信息系统组织争论的焦点，因为他们不同意一个特定的功能是对初始协议的改变还是部分改变。”范围变化的不可避免性使得我们需要一个正式的策略和过程来处理变化以及变化对进度和预算的影响。变化管理用来保护项目经理和团队，使其不会由于范围变化导致的进度和预算超支而困扰。

变化管理系统包括一组程序来记录变化请求，并根据变化的预期影响定义需要考虑的步骤。大多数变化管理系统要求一个或多个项目关联人员（例如，系统所有者、用户、分析员、设计人员或构造人员）使用变化请求表格。理想情况下，这些变化请求由变化控制委员会（Change Control Board, CCB）考虑，他们负责批准或者驳回所有的变化请求。CCB 的组成一般会包括项目团队的成员，也会包括对项目感兴趣或有利益关系的外部人员。CCB 的决策应建立在效果分析的基础之上。

可行性效果分析应该评估业务变化的重要性、变化对项目进度的影响以及变化对项目预算和长期运营费用的影响。

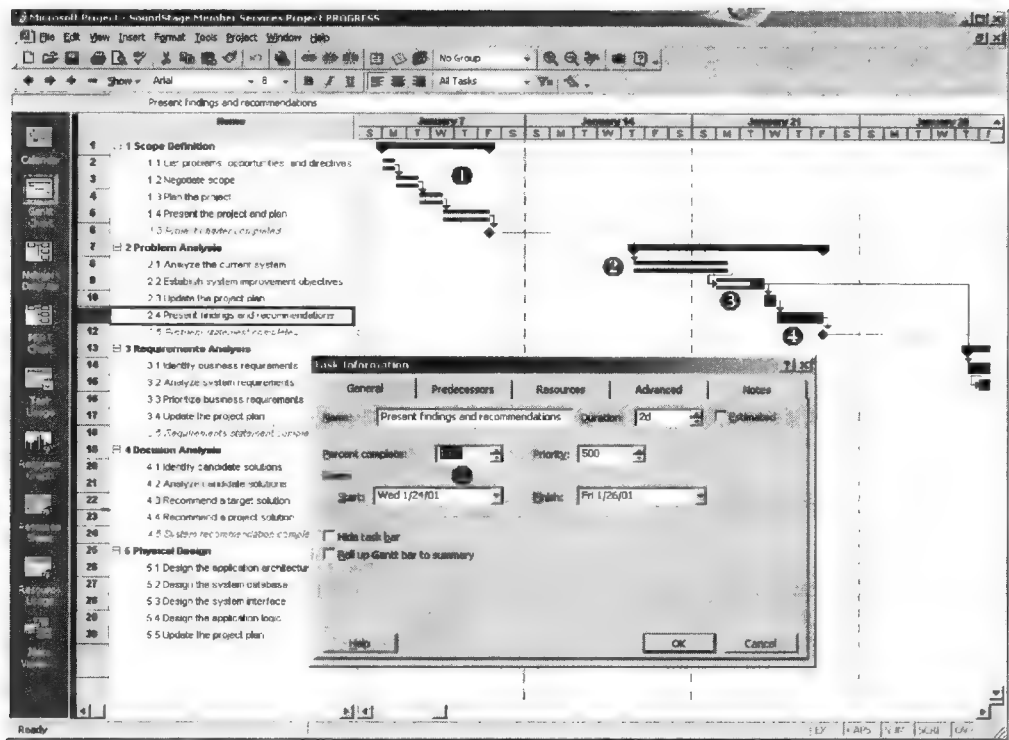


图 3-12 甘特图中的进展汇报

最后，变化管理重点在于管理客户的预期。在下一节中，我们将介绍一个简单但概念合理的框架，该框架用来管理预期以及预期对项目进度和预算的影响。

3.2.7.3 预期管理

有经验的项目经理经常抱怨管理系统所有者和用户对一个项目的预期比对费用、进度、人员或质量的管理都要困难得多。在本节中，我们介绍一个称为预期管理矩阵的简单工具，这个工具帮助项目经理处理管理问题。我们最初从 Phil Friedlander 博士那里接触到这个工具，Phil Friedlander 博士是麦道公司的顾问和培训员。他把这个矩阵说成是一种“民间传说”技术，但也归功于 Majer 的 Jerry Gordon 和 Ron Leflour（一位项目管理教育家和培训员）。Friedlander 博士的论文列在了本章的“推荐读物”中。为了便于使用，我们已经对这个工具做了小小的调整。

每个项目都存在目标和约束，表现为费用、进度、范围和质量。在理想情况下，你可以优化每个参数，管理层经常有这样的期望。但是，现实情况表明你不可能优化所有参数——你必须寻求一种对管理层来说既可行又可接受的平衡。这就是预期管理矩阵的目的。预期管理矩阵是一个规则驱动的工具，用于辅助管理人员理解变化的项目参数（例如费用、进度、范围和质量）的动态和影响。

基本的矩阵如图 3-13 所示，它包括 3 行和 3 列（加上标题）。行对应项目的成功度量：费用、进度以及范围和/或质量；列对应优先权：第一、第二和第三。为了便于建立预期，我们给优先权指定以下的名称：

- 最大或最小——对一个给定的项目来说，这被认为是最重要的成功度量。
- 受限的——在一个项目的三种成功度量中重要性居第二位。
- 可接受的——在一个项目的三种成功度量中是最不重要的。

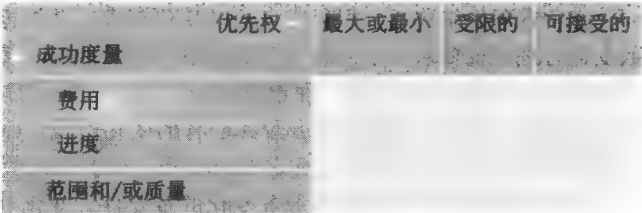


图 3-13 一个预期管理矩阵

大多数管理人员希望理想地对所有三个度量分配同样的优先权。经验表明三种度量会自然地趋于自我平衡。例如，如果你增加范围或质量需求，就需要花费更多的时间和/或费用。如果你想尽力使工作做得更快，一般来说则不得不削减范围或质量需求，或者支付更多的费用进行补偿。预期管理矩阵辅助（或者强制）管理人员使用三条简单的规则来理解这一点：

1. 对于任何项目，我们必须在9个可填的单元格中填上3个×。
2. 任何一行都不能超过一个×。即一个成功度量必须有且仅有一个优先权。
3. 任何一列都不能超过一个×。即必须各存在一个第一、第二和第三优先权。

让我们使用 Friedlander 博士自己的例子演示该工具的使用。在1961年，肯尼迪总统启动了一个大型项目——在10年内把人送到月球上并安全地返回地球。图3-14展示了这个项目实际的预期。让我们看一下这个例子。

1. 系统所有者（公众）既有范围预期，也有质量预期。项目范围（或需求）是成功地将人送上月球，项目质量度量是安全地将人返回。因为公众对这个新的太空计划的预期不会再少了，它必须作为第一优先权。换句话说，我们必须把最大的安全和最小的风险作为第一优先权。因此，在第1列第3行记录一个×。

2. 在项目启动时，当时的苏联已经走到了空间竞赛的前面。这是一个有关国家尊严的问题，所以，第二优先权给予了在10年内完成任务。我们称之为项目约束——没有必要提前最后期

限，但我们不想错过最后期限。这样，在第2列第2行记录一个×。

3. 默认情况下，第三优先权就是费用（1961年估计为200亿美元）。将费用标成第三优先权并不是说费用不用控制。我们只是说为了在约束的最后期限内实现项目范围和质量需求，可能不得不接受费用超支。在第3列第1行记录一个×。

历史记录显示我们实现了项目范围和质量需求，并且在1969年获得成功。这个项目实际上的花费超过了300亿美元——费用超支50%。费用超支使得项目失败了吗？相反，大多数人认为这个项目是一个巨大的成功。政府管理了公众的项目预期，即实现了最大的安全和最小的风险，而且满足了最后期限，这对于费用超支来说是一个可接受的权衡！政府明智地管理了公众的意愿。系统开发项目经理可以从这个平衡艺术中上一堂有价值的课。

在任何项目的开始阶段，项目经理应该考虑向系统所有者介绍预期矩阵的概念，并应该同系统所有者一起填写这个矩阵。对于大多数项目来说，很难在矩阵中记录下所有的范围和质量需求。相反，应该在工作陈述中列出。估计的费用和最后期限将直接记录在矩阵中。

假定你有了一个满足前面提到的规则的预期管理矩阵，它将如何帮助你管理预期？在一般的系统开发项目过程中，优先权是不稳定的，各种变化因素（例如经济、政府策略和公司策略）可以改变优先权：预算可能变得更受约束或者不那么受约束；最后期限可能变得更重要或不那么重要；质量可能变得更重要；并且，最经常发生的是需求增加了。正如已经指出的那样，这些变化因素以某种方式影响着所有的成功度量。解决问题的办法是管理预期，而不论项目参数如何变化。

这项技术是比较直观的。只要“最大/最小度量”或“受限的度量”变得不符合要求，就可能存在潜在的预期管理问题。例如，假设你正面对以下的优先权（见图3-15）。

1. 在项目开始阶段建立了明确的需求和质量预期并给予了最高优先权。

	优先权	最大或最小	受限的	可接受的
成功度量				
费用				×
• 200亿美元(估计)				
进度			×	
• 1969年12月31日(最后期限)				
范围和/或质量				
• 将人送上月球		×		
• 安全地将人返回				

图3-14 登月项目的预期管理

2. 为项目建立一个固定的最大预算。

3. 你同意力争希望的最后期限，但系统所有者接受了现实：如果必须把某些工作后延，可以对它进行调度。

现在假设在系统分析期间发现了重要的和未预料到的业务问题，对这些问题的分析将项目延迟到预期进度之后。而且，解决这些新需求最终扩展了对新系统的用户需求。作为项目经理，你将如何应对？首先，对进度的延误不要过度反应——进度延误在矩阵中具有“可接受的”优先权。项目范围的增加（以几个新需求的形式）是更重要的问题，因为增加的需求将导致增加项目费用。项目费用是受限制的成功度量。正如矩阵所指出的，我们有了一个需要解决的预期问题。是应该与系统所有者一起检查矩阵的时候了。

首先，系统所有者需要知道哪个度量或者哪些度量处于危险状态，以及为什么会处于危险状态。然后，项目经理和系统所有者可以一起讨论解决办法。可以采用以下几种解决办法：

- 可以重新分配资源（费用和/或进度）。也许系统所有者可以在其他地方找到更多的经费。所有的优先权将保持不变（注意：修改的最后期限是建立在系统分析过程中已经遇到的进度延迟基础上的）。
- 可以增加预算，但预算可以被计划外的进度延误抹平。例如，通过将项目延长到一个新的财政年度，就可以分配额外的经费，而不必从现有的项目经费中调配。这个方案如图 3-16 所示。
- 通过排列需求优先次序，并将某些需求推迟到系统的第 2 版实现，可以减少用户需求（或质量）。如果预算不能增加，这个替代方案将是合适的。
- 最后，可以改变度量优先权。

只有系统所有者才可以修改优先权。例如，也许系统所有者对增加的需求值得投入额外的经费表示同意。他（或她）分配足够的经费来满足需求，但同时调整优先权，以使最小费用成为最高优先权（见图 3-17，第 1 步）。但现在矩阵违反了一条规则——在第 1 列中有两个 ×。为此，我们必须把范围和/或质量准则调整到另一列中，在这个例子中，即“受限的”列（见图 3-17，第 2 步）。预期调整后，系统所有者冻结需求的增长并且仍接受进度延误。

关于优先权变化的问题还有最后三条说明。首先，在一个项目期间优先权可以多次变化。只要保持矩阵平衡（意味着它符合我们的规则），可以任意改变预期。其次，预期管理可以通过组合优先权变化和资源调整来实现。最后，即使项目符合进度，系统所有者也可以改变优先权。例如，在一个正在进行的项

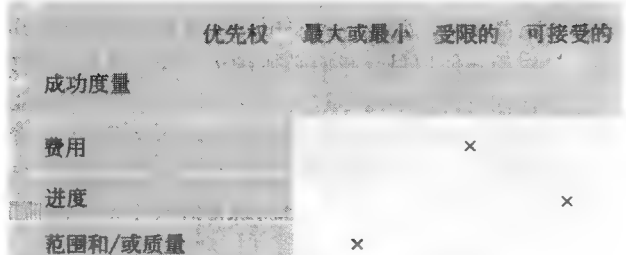


图 3-15 一个典型的预期矩阵

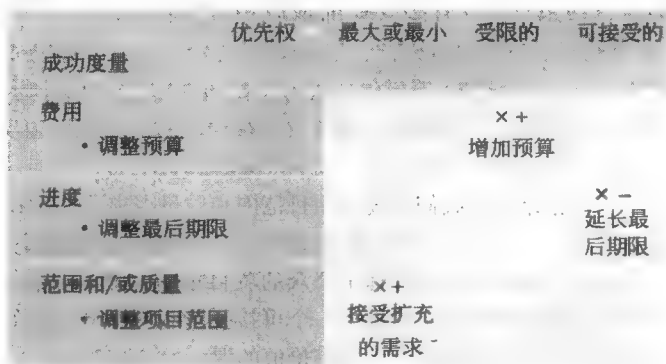


图 3-16 调整预期（一个例子）

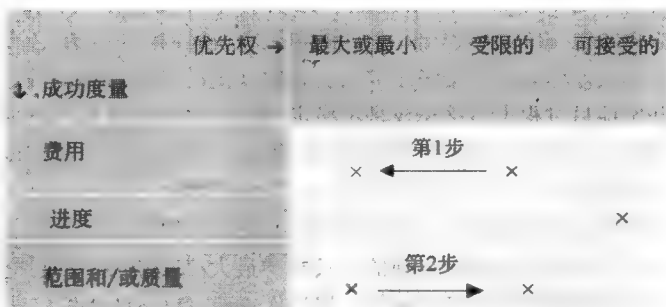


图 3-17 改变优先权

目中，政府规定可能会强制一个不可寻求折衷的最后期限。那将突然把我们的“可接受的”进度延误调整成为“最大约束”。另一个×将不得不被调整，以重新平衡矩阵。

预期管理矩阵是一个简单的工具，但有时简单的工具是最有效的工具。

3.2.7.4 进度调整——关键路径分析

当谈到项目进度时，某些任务对进度延迟比其他任务更敏感。因此，项目经理必须知道项目的关键路径和富余时间。

项目经理必须了解一个项目的关键路径和富余时间。这些项目因素影响着项目经理做出人员管理的决策。工作重点应该放在关键路径的任务上，而且如果需要，可以将资源临时地从具有富余时间的任务中转移出来，以帮助关键任务赶上进度。

项目的关键路径和富余时间可以从甘特图和 PERT 图中导出；但是，通常 PERT 图更常用，因为 PERT 图更清楚地描述了定义关键路径的任务之间的依赖关系。大多数项目管理软件（包括 Project）根据任务之间的依赖关系并组合任务工期自动地计算并加亮显示关键路径。但是，理解如何计算关键路径和富余时间仍是很有用的。

请考虑下面的例子。一个项目由 9 个主要任务构成，如图 3-18 所示。图中记录了每个任务的最可能工期（单位日）。这个项目中 4 个独立的任务序列，它们是：

路径 1：A→B→C→D→I

路径 2：A→B→C→E→I

路径 3：A→B→C→F→G→I

路径 4：A→B→C→F→H→I

每条路径最可能的总工期计算如下：

路径 1：3 + 2 + 2 + 7 + 5 = 19

路径 2：3 + 2 + 2 + 6 + 5 = 18

路径 3：3 + 2 + 2 + 3 + 2 + 5 = 17

路径 4：3 + 2 + 2 + 3 + 1 + 5 = 16

在这个例子中，路径 1 是关键路径，共 19 天（注意，如果它们具有同样的总工期的话，你可能会得到多条关键路径）。

在这个例子中，任务 E、F 和 G 不在关键路径上，它们都有富余时间。例如，任务 E 在一条比关键路径短一天的路径中，所以，任务 E 可以延误一天而不会影响项目完成日期。同样，任务 F 和任务 G 可以最多组合出两天富余时间，而不会延误整个进度。

在图 3-18 中，关键路径用粗线表示，具有富余时间的任务用细线表示。同样，项目管理软件也会在甘特图和 PERT 图中特别区分出关键路径任务。

3.2.8 活动 8——评估项目结果和经验

项目经理需要从错误中学习。他们要保持持续的过程改进。最后的这项管理活动是从项目团队成员（包括客户）获取有关的项目经验和反馈，旨在改进组织的项目管理和过程管理水平。为了回答以下基本问题，项目经理应该进行项目评审：

- 最终的产品是否满足用户预期或者超过了用户预期？
- 项目是否符合进度要求？
- 项目是否在预算范围内？

回答这些问题之后都应该紧接着问一个基本的问题：“为什么或者为什么没有？”最后，根据问题的回答，项目经理应该做出改变，以改进将用于未来项目中的系统开发和项目管理的方法。将改进建议要传送到“优化中心”，它可以相应地修改标准和过程以及同其他团队共享有用的想法和经验。通常项目评估能够对特定项目交付成果（里程碑）、产生这些交付成果的过程或任务以及项目的整个管理的改进做出贡献。

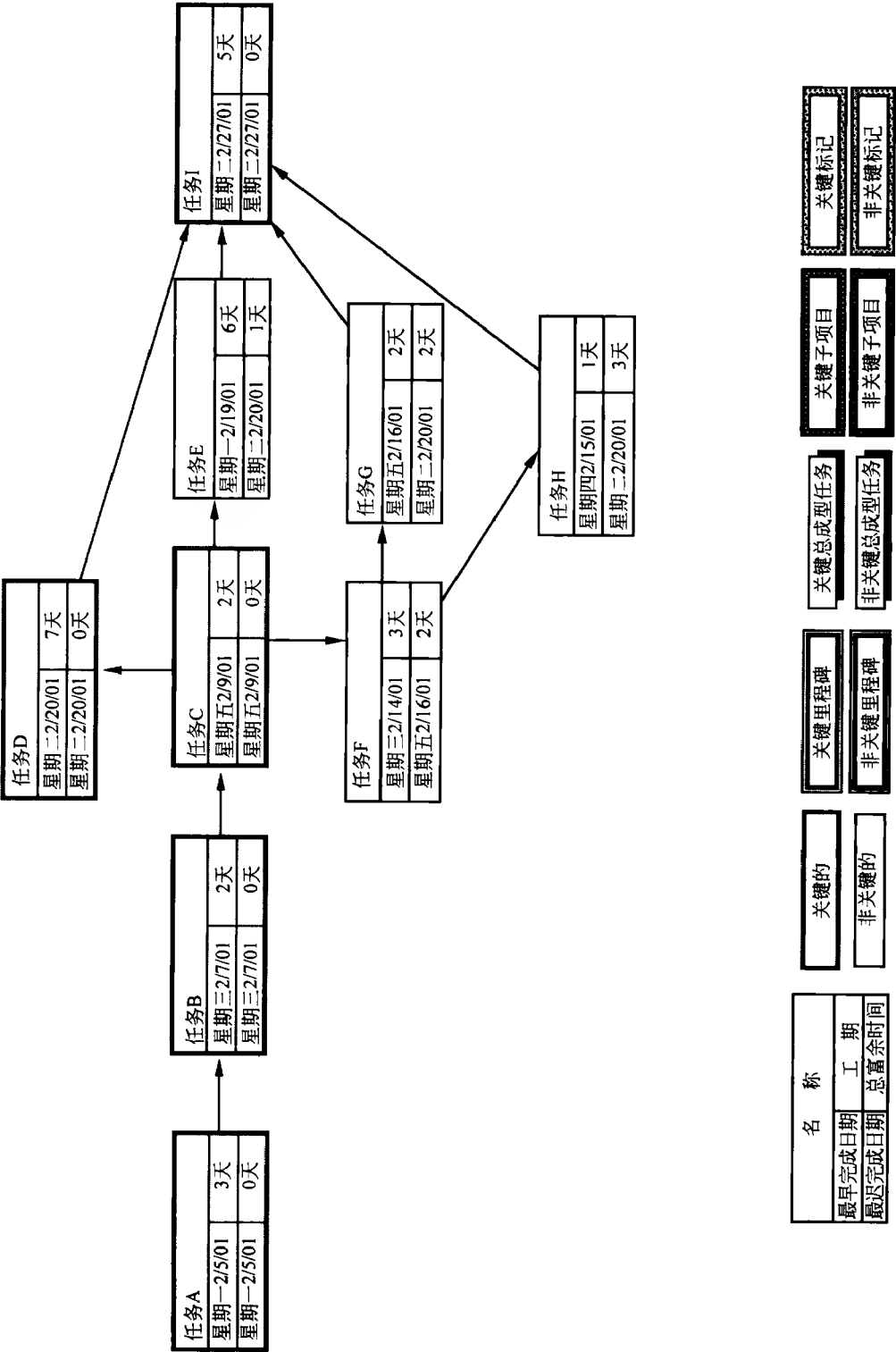


图3-18 关键路径分析

复习题

1. 什么是项目？
2. 在项目失败的各种不同原因中，项目失败的主要因素是什么？
3. 范围蔓延和特征蔓延之间有什么不同？
4. 项目经理应该具备的5种主要竞争力是什么？
5. 为什么业务实现竞争力很重要？
6. 基本的项目管理功能是什么？
7. 什么是 PERT 图和甘特图？如何决定使用哪一个？
8. 项目管理生命周期中的8个主要活动是什么？
 - 协商范围
 - 确定任务
 - 估计任务工期
 - 确定交叉任务依赖关系
 - 分配资源
9. 为什么协商范围很重要？在协商范围过程中有什么发布物？
10. 用于标示项目管理生命周期中的任务的流行工具是什么？
11. 在估计任务工期中要考虑哪些因素？
12. 正向调度和反向调度之间有什么区别？
13. 如何分类分配给项目的资源？
14. 项目经理应该如何管理项目期间产生的变化和/或需求的变化？
15. 为什么关键路径分析很重要？

问题和练习

1. 假设你是一位系统分析员，一个项目团队的高级成员，该项目团队刚好完成了一个大型项目，该项目持续了多年，而且几乎涉及了你所在企业的每个业务部门。项目正好在预算内提前完成。开发和实现进行得很顺利，几乎没有中断业务运行。实现后的一次调查表明系统用户经过很少的培训就能够使用该系统，但有更多的来自用户口头的评价认为该系统没有实现他们真正期望的，也没有实现该系统真正的用途。这个项目应该被认为是成功的项目吗？
2. 执行管理层开始关注有些用户对上一题中描述的新系统不太满意，并指派你领导一个实现后期工作组找出原因。在本书描述的十几个项目管理失误问题中，你认为最可能影响用户满意度的是哪些？
3. 作为一个新任命的经理，你渴望开始你的第一个项目。你的第一项活动应该是什么？它有多重要？它一般涉及什么人？你需要确切回答什么问题？这个活动的最终输出是什么，它的发布物包括哪些内容？
4. 你是一个中型项目的项目经理，这个项目计划从9月1日开始到第二年6月30日发布需要10个月的时间。现在是第二年的4月1日，项目已经进行7个月，项目稍微有点落后进度大约1周时间。绘制一张甘特图（你可以使用图4-2的风格，或者你喜欢的其他甘特图风格）。假设你使用的是 FAST 方法，而且项目阶段可以重叠。
 5. 你是一个公司的项目经理，该公司正在为你所在州的一些县设计行为健康系统。该项目稍微提前于进度，到目前为止还没有任何明显的问题。在检查准备构造的一些屏幕设计时，你吃惊地发现一些特征不是设计的内容。系统构造人员是公司一位最有才华和创造力的程序员。当你问到这些特性时，构造人员骄傲地告诉你这是他们增加到系统功能中的，没有花费额外编程时间，它们将是一个惊奇。你看到这些特征确实增加了系统的功能。代码已经写好，尽管它们不是被批准的技术设计的一部分，你应该允许它们被包含到系统中吗？
 6. 你的企业的方法学要求变更请求由一个变更控制委员会（CCB）考虑。在同程序员交流和讨论后，你决定提交一个变更请求给 CCB 以增加一个新特性。在给 CCB 的汇报中，你提出变更请求的原因是什么，你应该考虑哪些事情？
 7. 你的企业的 CEO 对你最近的项目的印象良好，以至于要你负责一个更大的，甚至更重要的项目。CEO 让你参加一个关于该项目重要性的讨论，告诉你企业的存亡寄托在这个项目的成败上，在预期竞争者会完成一个类似的项目之前给用户提供一个新系统。公司可以提供的预算只能达到一定的量，如果确实需要，也可以延期其他一些不太重要的正在进行的项目以获得一些额外的资金。最后，为了使其成为市场上的有竞争力的产品，新系统必须至少包含某个最小特征集，尽管希望拥有更多功能，但质量

是最重要的。在讨论结束时，CEO 握着你的手祝你好运。使用 CEO 设置的优先级创建一个初始预期管理矩阵。

- 8. 现在假设在这个项目过程中，显然费用被极大地低估了，预算快速地被耗尽。而且，市场部的领导拿到了一份商业杂志，读到了你的企业的主要竞争对手为它们的产品增加了一些确实令人激动的特性，而没有改变发布日期。预算超支不是主要问题，你知道还有其他的资金可用，尽管这会推迟其他项目。但你也知道市场部会要求在你正在开发的系统中增加对手产品的类似功能，并保持原来的进度。这代表了一个预期冲突，因为范围是成功的一个受限制的指标。这时你应该如何处理？
- 9. 假设 CEO 决定无论如何必须将新特征增加进来以保持新系统的竞争力。这会导致什么问题，如何在预期矩阵中反映？
- 10. 项目正按进度处于系统设计阶段，你正试图估计一个复杂的设计任务的工期。通过将这个任务分解成类似于你在其他项目中经历过的较小的任务，你估计这个任务正常的预期工期（ED）为 3 个工作日，假定 75% 的工作效率和 15% 的中断因素。但你也知道在一些情况下如果什么事情都不对时，需要用整整两周时间，或者最差工期（PD）80 个小时来完成这个设计。使用本书中介绍的典型技术，计算该任务最可能的工期。
- 11. 在上一题中，你使用什么技术来估计设计任务的预期工期？描述其他一些你可以用来估计任务工期的技术。

项目和研究

- 1. 项目失败，有时给人印象深刻。在网上搜索关于大型项目失败的文章。应该可以搜索得到大量的文章。查找并阅读大约 10 个关于最近 10 年来大型项目失败的文章，然后做如下研究：
 - a. 列举你找到的项目失败，并描述它们。
 - b. 每个项目失败的代价是什么？
 - c. 每个项目失败的后果是什么？
 - d. 基于本章列举的原因，对项目失败的原因进行分类。
 - e. 项目失败最常见的原因是什么？
 - f. 事后分析有多少项目失败是可以避免的？
 - g. 新的系统分析员可以从这些项目失败中学到的最重要的教训是什么？

- 12. 在项目的某个阶段，你检查了项目进度，你发现项目团队中的某个成员被分配了多个任务，这些任务加起来比这期间这个人可以工作的时间要长几个小时。你能用什么技术来解决这个问题？
- 13. 你被要求用最短的时间来完成一个任务。项目任务、最可能工期（单位天）和前提任务如下。有哪些不同的路径（任务序列），每个路径的天数是多少？哪条是关键路径，即项目可以完成的最短时间？在商业世界中项目经理对关键路径分析的理解是否真的很重要，还是只是理论知识？

任务	工期	前提任务
A	2	无
B	2	无
C	1	无
D	4	A
E	5	B
F	1	C、D
G	6	A、E
H	4	F
I	7	G、H

- 14. 作为一个快速成长的企业的新的项目经理，你被要求领导一个项目团队完成一个重要的项目。项目的范围不太明朗，项目时间周期有点紧但还是可行的，预算很充足。事实上，你被授权你想要多少人参加项目就可以雇用多少人。你估计 5 个人应该正好够这个项目，8 个人则可以提供较好的备份余量，10 个人则可以在要求的时间内完成一个出色的系统的资源。在你决定雇用多少人之前，你应该记住什么？

- 2. 项目管理研究所（PMI）是世界上领导地位的项目管理组织之一，也许是唯一的。PMI 创建和维护着“项目管理知识集”（PMBOK），它是项目经理们事实上的标准。PMI 也为提交具有规定的知识和经验的项目经理进行“项目管理职业”（PMP）认证。
 - a. 访问 PMI 网站（www.pmi.org）。获得 PMP 认证有什么要求？
 - b. 根据你的阅读和经验，你认为 PMP 认证对于项目经理有多重要？你认为值得投资进行认证吗？
 - c. 雇用 PMP 的企业如何呢？认证能保证企业的项目成功完成吗？企业应该给一个具有 PMP

- 认证的项目经理多高的待遇？
- d. 在许多领域的从业人员需要被认证，例如医师、工程师、会计师和律师。你认为在一个人能够管理一个大型项目前应该要求具有职业证书吗？
 3. 你在一个大型法律公司的信息技术部门工作，该公司在全州都设有办事处。公司的副总裁要求你为公司管理一个自动化案件跟踪系统的开发。这个项目刚刚开始，是你被要求管理的第一个大型项目。你很认真地履行你的职责，想把这个项目做得很出色。
 - a. 你正在同公司的副总裁会讨论项目的范围。在会上，需要回答和协商什么问题，以便能够确定项目的范围？
 - b. 完成范围协商之后，副总裁要求你写一份工作陈述。在这种情况下，工作陈述代表了什么？应该写多长篇幅？
 - c. 使用如图 3-5 所示的大纲作为例子，写一份工作陈述。假设副总裁给了你全权委托（尽管现实生活中从不会发生）。
 4. 项目管理软件，例如微软的 Project，已经变得很常见。许多工具包含了传统的工具，例如 PERT 图和甘特图，这些工具是几十年前开发的。
 - a. 对十几个企业的项目经理进行一次非正式的调查。他们中有多少人使用项目管理软件？
 - b. 对于那些不使用项目管理软件的项目经理，他们不使用的原因是什么？
 - c. 对于那些使用项目管理软件的项目经理，他们使用哪个软件？他们对所使用的软件有什么看法？
 - d. 在网上查找项目管理软件，你找到了哪些软件？
 - e. 了解这些软件的特征和说明。它们呈现出同样的特征吗？你认为哪一个最流行，或者最广泛使用？如果不考虑费用的话，你将选用哪一个？
 5. 你正在为你所在的大型律师事务所管理一个案件跟踪系统的开发。项目的需求阶段基本上结束了，初始设计工作已经开始。项目比进度延迟了几天，你认为这并不严重，项目仍在预算之内，但刚刚好。按照需求分析，质量问题到目前为止按你的观点基本可接受，但一些项目成员提出他们并不确信某些问题完全被解决了。根据这些信息，写一份项目进展报告，使用图 3-11 的提纲作为例子，并遵循本章活动 7 描述的指南。
 6. 作为持续改进的一部分，对项目经理和项目团队来说，当项目结束时评估结果和经验很重要。可以用许多方法和技术进行这份工作。从网上查找有关的文章，使用项目评估、项目实现后报告等等之类的关键词。
 - a. 你找到了什么文章？
 - b. 描述文章中建议的方法和技术。
 - c. 选择你觉得最有价值的几篇，并解释原因？
 - d. 你认为评估项目结果会对未来项目的结果产生很大的影响吗？

小型案例

1. 你的团队正在为某个本地的客户卡车修理（Custom Car Care）公司开发 Web 网站。需求分析、开发和成功部署的进度固定为 4 个月。团队按进度在第 8 周刚给客户（卡车修理公司的 CEO Debbie）展示了原型系统。Debbie 对你们目前的工作很满意，但想增加一些额外的功能。尽管增加的功能没有进行预期的时间或费用估计，但她要求你们仍保持进度并在预算内完成。你应该怎么做？
2. Alicia 和 John 是一个用 Java 编写大程序困难的团队。他们对这种语言有一些经验，但仍不得不“边干边学”很多东西。他们估计这个项目最优工期要花 2 个月的时间，预期工期是 3 个月，最差估计是 4 个月。你是他们的项目经理，要同客户商讨一个关于代码开发完成的合同。你应该在合同中为这个发布物留出多少时间？
3. 为你正在课堂上完成的某个大项目同时进行任务和时间正向调度和反向调度。描述这两种调度的结果有什么差异。当你完成项目时，监督你的项目时限并跟踪里程碑。在项目结束时，提交你的项目时限和项目记录给教授，附带一份项目的副本。项目进度开发和管理对你有帮助吗？与你的同学分享你的经验。
4. 在同一个项目经理的交谈中，找出影响项目成功（按时）完成的个人因素有多大。这个项目经理是如何处理团队关键人员的个人和家庭问题（这些问题可能导致关键人员分心或离开）？

团队和个人练习

1. 对指导老师：创建一个4人的团队，并指定一个人作为项目经理。给他们分配一个具有很短时限的挑战性的任务。任务应该是对他们可行的，但确实不容易。在项目的中期，每个团队交换一个成员，以使每个团队都损失一名成员，并获得另一个新成员。不允许团队同那个“被解雇”的成员交谈。
项目经理记录下他们是如何处理这种情况了吗，出现了什么问题，他们将如何处理一个在未来不

同的团队（知道他们会在任何时候没有任何提示的情况下损失一个成员）。

2. 团队或个人练习：对每个课堂项目，开发一个计划。记录下你能够想到的每件可能出错的事以及一个连续的项目计划。
3. 作为一个团队，出去吃午饭或晚饭。分享一些团队可能不了解的你生活中的观点。找出一些你所不知道的每个成员的事情。

系统分析方法

第二部分用 7 章篇幅介绍系统分析活动和方法。第 4 章（系统分析）通过介绍系统分析活动为所有后续章节提供基础。系统分析阶段是一个项目中最关键的阶段。在系统分析期间，需要了解现有业务系统，理解其中的问题，定义改进目标，并确定后续技术方案必须实现的详细业务需求。很明显，一个新系统设计和实现的质量在很大程度上取决于系统分析的质量。在一个项目中，系统分析经常被简化了，这是因为：1）许多分析员对要使用的概念和逻辑建模技术不熟练；2）许多分析员不清楚缺少系统分析的严重后果。第 4 章概述系统分析及其在项目中的重要性。后续的章节讲授专门的系统分析技术，重点强调逻辑系统建模技术。

第 5 章（需求获取的调查研究技术）讲授用于征求新系统用户需求的各种调查研究技术。

第 6 章（使用用例建模系统需求）介绍使用用例建模技术记录系统需求的工具和技术。

第 7 章（数据建模和分析）讲授数据建模技术——一种为系统组织和记录数据存储需求的技术。你将学会绘制实体关系图，实体关系图用来结构化最终将被设计成数据库的业务数据。这些模型将收集控制数据的业务关联和规则。

第 8 章（过程建模）介绍过程建模技术。这一章解释如何使用数据流图描述系统中基本的业务过程（采用的方法是绘制系统的数据流）以及由过程实现的策略和程序。如果你已经从事过一些程序设计工作，那么你将认识到理解业务过程对于编写程序来说是十分重要的。

第 9 章（使用 UML 进行面向对象分析和建模）讲授使用 UML 工具进行系统分析的面向对象方法。

第 10 章（可行性分析和系统方案建议）讲授如何集体讨论可能的系统方案，分析这些方案的可行性，选择整体最优的方案，然后以书面和口头建议的形式向管理层展示你的建议。

系统分析

本章概述和学习目标

在本章中，你将更深入地了解系统开发项目中的系统分析阶段，即：范围定义阶段、问题分析阶段、需求分析阶段和决策分析阶段。前三个阶段合在一起称为系统分析，最后一个阶段涉及系统分析和系统设计之间的转换。本章将介绍以下内容：

- 定义系统分析并将这个定义与本书的系统开发方法学的范围定义阶段、问题分析阶段、需求分析阶段、逻辑设计阶段和决策分析阶段联系起来。
- 描述一些用于解决企业系统问题的系统分析方法。
- 按照信息系统构件描述范围定义阶段、问题分析阶段、需求分析阶段、逻辑设计阶段和决策分析阶段。
- 按照目的、参与者、输入、输出、技术和步骤等内容描述范围定义阶段、问题分析阶段、需求分析阶段、逻辑设计阶段和决策分析阶段。
- 确定本书中有助于学习专门的系统分析工具和技术的章节和模块。

注意 尽管本章介绍了一些系统分析的工具和技术，但本章的目的并不是讲授这些工具和技术。本章仅仅讲授系统分析的过程，工具和技术将在后面6章中讲解。

本章关键术语

系统分析 (systems analysis) 是一种问题解决技术，它将一个系统分解成各个组成部分，目的是研究各个部分如何工作、如何交互，以实现其系统目标。

系统设计 (systems design) 是一种互补的问题解决技术（对于系统分析来说），它将系统的组成部分重新装配成一个完整系统——希望得到一个改进的系统。这可能包括增加、删除和改变原始系统的某些部分。

信息系统分析 (information systems analysis) 定义为一个信息系统开发项目中的这样一些开发阶段，这些阶段的重点是业务问题和需求，这些需求独立于实现方案中可能使用的任何技术。

资料库 (repository) 是系统分析员、系统设计人员和系统构造人员保存与一个或多个系统或项目有关的文档的地方。

模型驱动分析 (model-driven analysis) 强调绘制图形化系统模型来记录和验证现有的和/或建议的系统。系统模型最终将成为设计和构造一个改进的系统的蓝图。

模型 (model) 是对现实或构想的一种表述。因为“一幅图胜过千言万语”，所以大多数模型使用图形方式表述现实或构想。

结构化分析 (structured analysis) 是模型驱动的，以过程为中心的技术，用于分析一个现有系统，定义新系统的业务需求，或者同时用于这两种用途。模型是展示系统组件的图形，内容包括过程及其相关的输入、输出和文件。

信息工程 (Information Engineering, IE) 是一种用来计划、分析和设计信息系统的模型驱动的、以数据为中心的但对过程敏感的技术。IE模型是一些说明和同步系统的数据和过程的图形。

对象 (object) 封装了描述离散的个人、对象、地点、事件或事物的数据（称为属性）以及所有使用或修改数据和属性的过程（称为方法）。访问或修改对象的数据的唯一方法是使用对象预定义的过程。

面向对象方法 (object-oriented approach) 是一种模型驱动的技术，它将数据和过程集成到被称为对象的结构中。对象模型是从各个方面（例如结构和行为以及对象的交互）说明系统的对象的图形。

原型 (prototype) 是一个预期系统的小规模的、不完整的但可工作的示例。

获取原型 (discovery prototyping) 向用户提供响应需求的一个快速而粗略的实现，以确定用户的业务需求。

快速架构分析 (rapid architected analysis) 试图从现有系统或获取原型中导出系统模型。

逆向工程 (reverse engineering) 技术读取一个现有数据库、应用程序或用户界面的程序代码，并自动生成等价的系统模型。

需求获取 (requirements discovery) 包括系统分析员用来从用户团体那里确定或提取系统问题和方案需求的那些技术。

联合需求计划 (Joint Requirements Planning, JRP) 通过研讨会将所有的系统所有者、系统用户、系统分析员和一些系统设计人员及构造人员组织在一起，进行系统分析。JRP 一般被看作是联合应用开发 (JAD) 的一部分，JAD 是一种更全面的应用 JRP 于整个系统开发过程的技术。

调查研究 (fact-finding) 技术用于收集有关系统问题、机会、方案需求和需求优先权的信息。也称为信息收集。

业务过程重构 (Business Process Redesign, BPR) 是系统分析方法的应用，目标是独立于信息技术动态地改变和改进一个组织的基本业务过程。

敏捷方法 (agile method) 集成各种系统分析和设计方法，根据要解决的问题和要开发的系统应用合适的方法。

指导部门 (steering body) 是一个由主要业务管理人员和系统管理人员构成的委员会，它研究相互竞争的项目建议，为项目排列先后次序，确定哪个项目将为组织带来最大的价值，并因此应该被批准继续进行系统开发。它也称为指导委员会 (steering committee)。

因果分析 (cause-and-effect analysis) 是一种研究问题以确定其原因和结果的技术。

目标 (objective) 是一种对成功的度量。它假定在有足够资源的条件下希望实现的某些东西。

约束条件 (constraint) 是将限制你灵活地为目标定义方案的某些东西。约束条件基本上无法被改变。

功能需求 (functional requirement) 描述一个系统必须提供的活动和服务。

非功能需求 (nonfunctional requirement) 描述一个满意的系统的其他特征、特点和约束条件。

用例 (use case) 是业务场景或事件，系统必须对这些场景或事件提供确定的响应。用例来自面向对象分析中，但它们在许多系统分析和设计方法中都很常见。

时间盒 (timeboxing) 技术是一种以版本化的形式发布信息系统功能和需求的技术。开发团队选择系统的最小子集，这个子集如果完全实现，就能够立即向系统所有者和用户返回价值。理想情况下，子集应该在 6~9 个月或更少的时间内开发出来；之后，系统的增值版本以类似的时间段开发出来。

4.1 什么是系统分析

第2章介绍了系统开发过程，并简单讨论了开发过程的各个阶段。在本章中，我们将更详细地讲解那些共同构成系统分析的阶段。系统分析是对系统及其构成部分进行研究，作为系统设计的前提条件（新系统和改进系统的规格说明）。本章重点是系统分析，第11章则重点介绍系统设计。

系统分析更现代一点的定义是：系统分析是一个描述系统开发各个早期阶段的词汇。图4-1标识出FAST方法学（见第2章）的完整典型开发路线中的系统分析阶段。实际上，不存在一个被广泛接受的信息系统分析定义，也从来没有就什么时候系统分析结束和什么时候系统设计开始达成过广泛一致的意见。但为了便于叙述，信息系统分析强调业务问题方面，而非技术或实现方面。

“系统所有者”和“系统用户”对业务的考虑推动了系统分析工作的进行。因此，系统分析涉及从系统所有者和系统用户角度看待的“知识”、“过程”和“通信”构件，“系统分析员”则是系统分析的推动者。这些内容在本章前面的“主页”中已经说明。

系统分析产生的文档和交付成果通常存储在资料库中。可以为单个项目建立一个资料库，也可以使用所有项目和系统共享一个资料库。资料库通常包括以下内容：

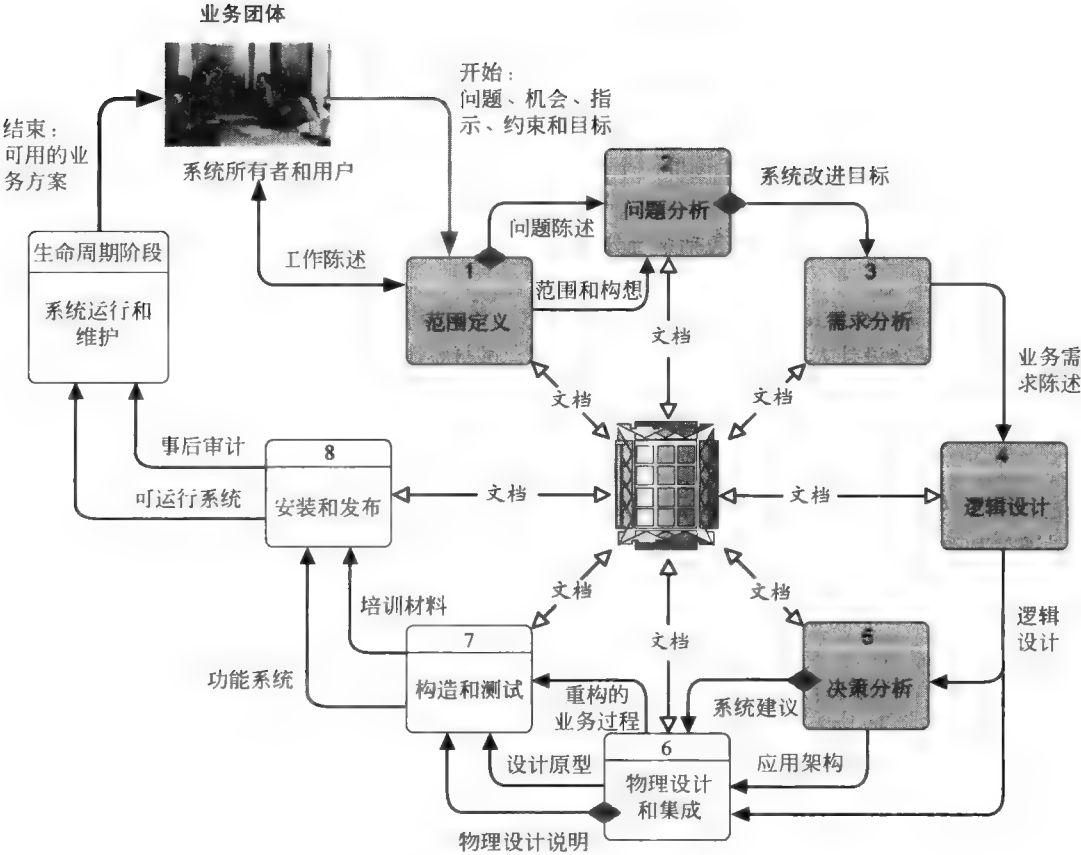


图 4-1 系统分析上下文

- 一个网络目录，目录存储了字处理软件、电子表格软件以及其他含有项目信件、报告和数据的计算机文件。
- 一个或多个 CASE 工具目录或百科全书（见第 2 章）。
- 打印的文档（例如存储在活页夹和系统库中）。
- 一个链接到上述组件的内联网网站接口（用于沟通）。

从本章起，我们将这些组件总称为资料库。

本章将更详细地分析这 5 个系统分析阶段。首先，介绍一些系统分析的整体策略。

4.2 系统分析方法

系统分析的基础是问题解决技术。由于解决问题的方法很多，所以系统分析方法也就有很多。较流行的系统分析方法是结构化分析、信息工程、获取原型和面向对象分析，这些方法经常被看作是相互竞争的可互相替代的技术。但实际上，这些方法可以互补，这在第 2 章中称为敏捷方法。我们将简要地介绍这些方法。

注意 本章的目的仅仅是在高层次上理解方法，本部分中的后续各章节将讲授这些基本技术。

4.2.1 模型驱动分析法

结构化分析、信息工程和面向对象分析是模型驱动分析的例子。模型驱动分析使用图形交流业务问题、需求和方案。读者可能已经熟悉的模型例子有：流程图、结构或层次图和组织结构图。

模型驱动分析法是第 2 章中介绍的模型驱动方法学和开发路线的特色。下面简单介绍三种最流行的模型驱动分析法。

4.2.1.1 传统方法

20 世纪 70 年代初期开发了许多传统的系统分析和设计方法。结构化分析是最早的正式信息系统分析方法之一，目前仍是最广泛应用的方法之一。结构化分析关注数据通过业务和软件过程的流程，又称为以过程为中心的。以过程为中心是指这项技术强调的是信息系统框架中的“过程”构件。

数据流图（见图 4-2）是用于建模过程的关键工具之一，数据流图描述在一个系统中现有的和/或建议的过程，以及它们的输入、输出和数据。模型描述了过程之间和通过过程的数据流，并展示了数据存储在哪里。最后，这些过程模型作为被实现的业务过程以及被购买或构造的计算机程序的蓝图。

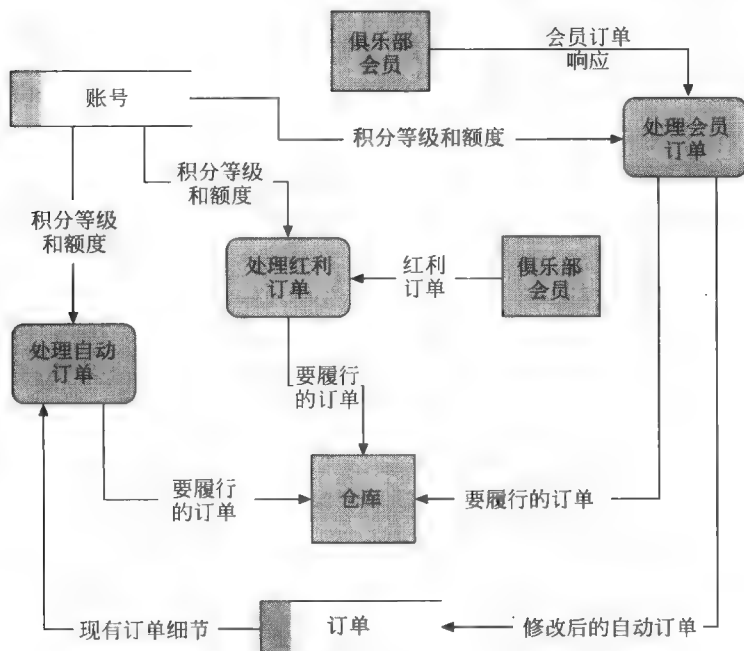


图 4-2 一个简单的过程模型（也称为数据流图）

由于面向对象方法的流行，软件设计的结构化分析技术已经基本上不再使用了。但如今，过程建模技术又获得了重视，这要归功于业务过程重构，将在本章后面详细讨论业务过程重构法。

另一种传统方法，称为信息工程（IE），它关注系统中存储的数据结构，而不是过程。因此，又称为以数据为中心的，强调对“知识”（或数据）需求的分析。实体关系图（见图 4-3）是建模数据需求的关键工具，实体关系图仍广泛用于设计关系数据库。

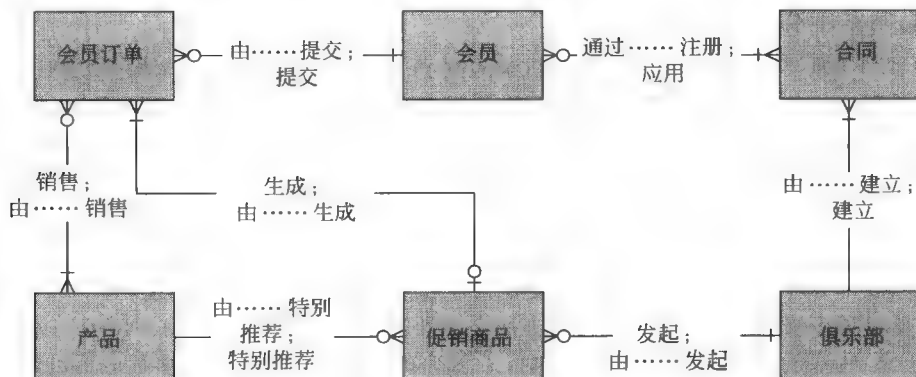


图 4-3 一个简单的数据模型（也称为实体关系图）

原先将信息工程看作是同结构化分析竞争的方法。但随着时间的推移，人们将它们进行了互补：使用数据流图建模系统过程，使用实体关系图建模系统数据。

4.2.1.2 面向对象方法

传统的系统开发方法有意地分开考虑“知识”（数据）和“过程”方面。尽管许多系统分析方法已经做了大量的努力试图使数据模型和过程模型同步，但收效甚微。因此对象技术兴起，以消除这种数据和过程的人为分离现象。面向对象方法不把信息系统看作数据和过程，而是一组封装了数据和过程的对象。对象可以包含数据属性。但是，创建、读取、修改或删除对象的数据的唯一方式是通过嵌入对象的过程（称为方法）进行。面向对象编程语言变得越来越流行，例如 Java、C++ 和 .NET 语言。

面向对象方法拥有一整套称为统一建模语言（UML）的建模工具。图 4-4 展示了 UML 的一种图形——对象类图。有些 UML 工具在系统项目中获得了接受，即使信息系统并不是由面向对象技术来实现的。

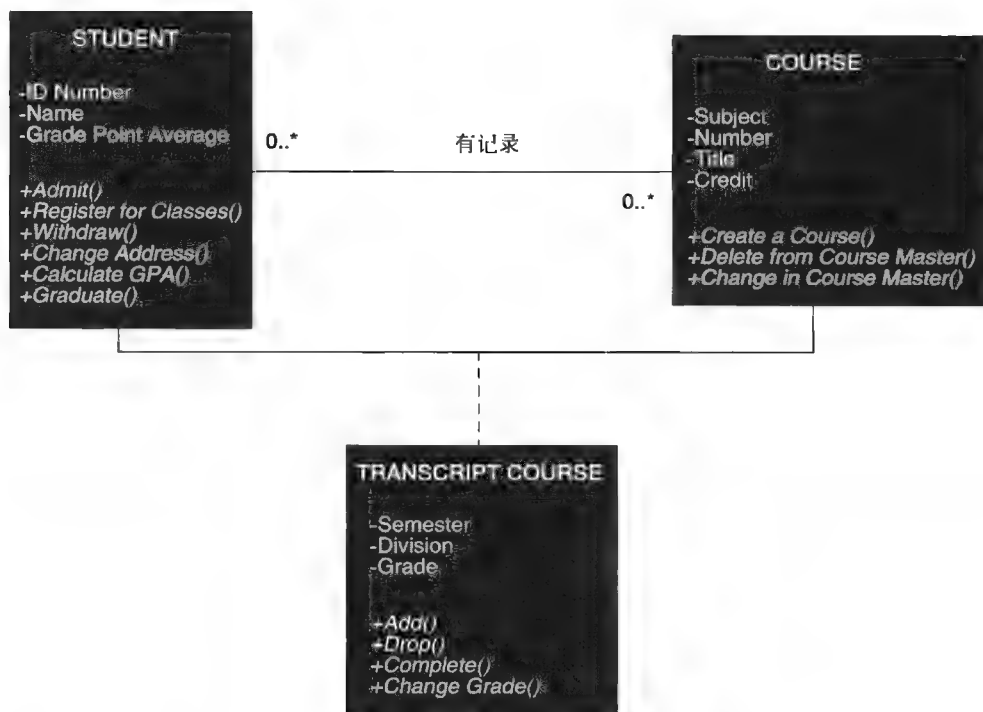


图 4-4 一个对象模型（使用统一建模语言标准）

4.2.2 加速系统分析法

获取原型和快速架构开发是加速系统分析法的例子，加速系统分析法强调构造原型以便更快速地为一个新系统确定业务需求和用户需求。大多数这类方法从构造原型的某些变种导出，原型是一个预期系统的小规模的、不完整的但可工作的示例。原型迎合了“当我看到它时，我才知道我想要什么”的思维方式，这种思维方式是许多用户和管理人员的特点。“不完整”是指原型系统将不包括一个完善的应用系统应该具有的错误检查、输入数据验证、安全和过程完整性。它既不会被打磨精化，也不会像一个最终系统一样为用户提供帮助。但因为它可以被快速地开发出来，所以它可以快速地确定业务级需求的最重要部分。有时原型系统也可以发展成为完整的信息系统和应用系统。

这些方法常见于快速应用开发（RAD）方法学和开发路线（参见第 2 章）。RAD 方法需要自动化工具的支持。一些基于资料库的 CASE 工具包括了很简单的 RAD 工具，但大多数分析员使用各种实用的 RAD 编程环境，例如 Sybase 的 PowerBuilder、Microsoft 的 Access、Visual Basic.NET 或 IBM 的 Web-sphere Studio for Application Development (Java-based)。

下面简单介绍两种目前流行的加速分析法。

4.2.2.1 获取原型

获取原型使用快速开发技术辅助用户获取业务需求。例如，对于系统分析员来说，以下情况很常见：使用一个简单的开发工具（例如 Access）快速地构造一个简单的数据库，用户输入表单和示例报告，然后征求用户的响应，看这个数据库、表单和报告是否表示了用户的业务需求。一般来说，使用一个较复杂的应用开发工具和语言开发最终的新系统，但简单的工具使得系统分析员可以更快地建立用户需求的原型。

在获取原型中，我们不鼓励让用户担心原型最终的外观（look and feel）——那些都可以在系统设计期间改变！这也正是对原型系统的主要批评——原型工具中的软件模板可以快速地生成很雅致并且视觉效果很吸引人的原型系统，这可能会鼓励过早地关注和投入原型所表现出来的设计。用户也可能被误导而认为：1) 可以这样快速地建成完整系统；2) 可以用 Access 工具来构建最终系统。

无论如何，获取原型仍是一种首选和推荐的方法。遗憾的是，有些系统分析员和开发人员正使用获取原型完全地替代模型驱动设计，这样做只会学到真正的工程师已经懂得了多年的道理——如果没有一定量的更正式的设计，你无法构造原型。下面介绍快速架构分析。

4.2.2.2 快速架构分析

快速架构分析也是一种构建系统模型的加速分析法。快速架构分析由于逆向工程技术而成为可能，许多自动化工具中包含逆向工程技术，例如 CASE 和编程语言（参见第2章）。逆向工程工具试图从现有系统或获取原型中导出系统模型（如本节前面描述的）。系统分析员和用户可以编辑和改进得到的系统模型，然后为一个改进的和新的系统提供蓝图。快速架构分析是模型驱动法和加速分析法的混合。

有两种不同的技术应用于快速架构分析：

- 大多数系统已经进行了某种程度的自动化并作为遗留信息系统而存在。许多 CASE 工具可以读取底层的数据库结构和/或应用程序，并对其进行逆向工程，导出各种系统模型。这些模型作为模型驱动的用户需求分析的出发点。
- 如果原型已经在 Microsoft 的 Access 或 Visual Basic 之类的工具中建立，这些原型有时可以被逆向工程为等价的系统模型。系统模型通常更容易用于分析用户需求的一致性、完整性、稳定性、可伸缩性和对未来变化的灵活性。而且，系统模型常常可以使用同样的 CASE 工具和 ADE 正向工程为使用更健壮的企业级数据库和编程技术的数据库和应用程序的模板或框架。

这两种技术都涉及到前面的问题，即：在完全缺少一个更正式的设计的情况下，工程师难以进行原型化，同时还要保持加速系统分析阶段的优点。

4.2.3 需求获取法

模型驱动法和加速系统分析法都试图表达对一个新系统的用户需求，或者使用模型或者使用原型系统，但这两种方法都需要依赖于需求的确定和管理。更进一步说，系统需求依赖于发现当前系统中存在的问题和机会；因此，分析员必须在确定问题、机会和需求方面很熟练。所以，系统分析的所有方法都需要某种形式的需求获取。下面简单介绍两个常用的需求获取法。

4.2.3.1 调查研究技术

调查研究技术是所有系统分析员都应该掌握的一项基本技能。本书中介绍的调查研究技术（在下一章介绍）包括：

- 对现有文档、报告、表单、文件、数据库和备忘录进行抽样。
- 研究相关文献、权衡其他方案和实地考察。
- 观察当前系统的运转和工作环境。
- 调查和咨询管理人员和用户团体。
- 同合适的管理人员、用户和技术人员面谈。

4.2.3.2 联合需求计划

上面列出的典型的调查研究技术是很重要的，但是可能很耗时。**联合需求计划**（JRP）技术可以明显

加速需求的获取和管理。一位受过 JRP 培训或认证的分析员通常扮演着研讨会主持人的角色，研讨会一般需要 3~5 天的全天的工作日。这个研讨会可以代替几个星期或几个月的传统调查研究及后续会议。

JRP 提供了一种加速所有系统分析任务和交付成果的工作环境，它促进了系统所有者和系统用户更积极地参与到系统分析中。但它也要求主持人具有较高的协调和协商能力，以确保所有与会各方得到适当的机会为系统开发做出贡献。

JRP 通常和前面描述的模型驱动分析方法一起使用，并且一般被包含在第 2 章中介绍的快速应用开发（RAD）方法学和开发路线中。

4.2.4 业务过程重构法

现代系统分析方法的一个有趣的应用是**业务过程重构（BPR）**。对 BPR 的兴趣是由下述因素所驱动的：大多数现有信息系统和应用系统仅仅是自动化了原有的低效业务过程。自动化的官僚主义仍然是官僚主义；自动化并不会给企业创造价值，而且实际上可能会适得其反。在第 1 章中介绍的 BPR 是由全面质量管理（TQM）和持续过程改进（CPI）引发的许多项目中的一个。

有些 BPR 项目的重点是所有的业务过程，而不考虑自动化程度。BPR 要求全面地研究和分析每个业务过程的瓶颈、价值回报以及取消或理顺的机会。过程模型（例如数据流图）帮助企业可视化它们的过程。一旦重新设计了业务过程，大多数 BPR 项目就检查如何最佳地应用信息技术改进业务过程。为了实现或支持新的业务过程，可能会产生新的信息系统和应用系统开发项目。

BPR 也应用于信息系统开发项目中，经常用来研究现有业务过程以确定问题、官僚主义和低效率出现的地方，这些问题可能被改进的新信息系统和计算机应用系统的需求所涉及。

BPR 在基于购买和集成商用现成产品（COTS）的 IS 项目中也常见。COTS 软件的购买通常要求一个企业调整其业务过程以适应该软件。在系统分析期间对现有业务过程的分析通常是这类项目的一部分。

4.2.5 系统分析策略

同大多数商用方法一样，我们在本书中使用的方法在系统分析上并不强制使用某种单一方法。相反，它集成了前面介绍的所有流行方法，成为一套**敏捷方法**。音阶公司案例研究将在一个典型的系统分析员首次任职的背景下演示这些方法。系统分析技术将用于该框架的以下内容：

- 信息系统构件。
- 开发阶段。
- 实现一个开发阶段的开发任务。

给定用于研究系统分析的上下文后，我们现在可以探索系统分析的开发阶段和开发任务。

4.3 范围定义阶段

如第 2 章所述，范围定义阶段是典型系统开发过程的第一个阶段。在其他方法学中，这个阶段可能叫做**初始研究阶段**、**开始研究阶段**、**概述阶段**或**计划阶段**。范围定义阶段回答这样一个问题：“这个项目看起来是否值得？”为了回答这个问题，我们必须定义项目的范围以及触发该项目的可见的问题、机会和指示。假设项目看起来值得考虑，该阶段就需要按照范围、开发策略或“开发路线”、进度、资源需求和预算制定项目计划。^①

图 4-5 是本章的 5 个任务图中的第一个，我们将在本章介绍这 5 个任务图，详细地查看系统分析的每个阶段。任务图显示了为完成一个阶段要实施的工作（任务）。我们的任务图并不强制采用任何专门的方法，但我们将在相应的段落中描述你可能会在每个任务中考虑使用的那些方法、工具和技术。图 4-5 显示了范围定义阶段需要完成的任务。注意，这些任务图仅仅是模板，项目团队和项目经理可以扩展这些模板，也可以修改这些模板以反映任何给定项目的特定需要。

① 如果你已经阅读过第 3 章，应该意识到这些计划要素是项目管理的一部分。第 3 章概述了项目经理制定项目计划的过程。

SoundStage Entertainment Club Information System Services Phone: 494-0666 Fax: 494-0999 Internet: http://www.soundstage.com Intranet: http://www.soundstage.com/iss		REQUEST FOR INFORMATION SYSTEM SERVICES	
DATE OF REQUEST		SERVICE REQUESTED FOR DEPARTMENT(S)	
January 9, 2003		Member Services, Warehouse, Shipping	
SUBMITTED BY (key user contact) Name Sarah Hartman Title Business Analyst, Member Services Office B035 Phone 494-0867		EXECUTIVE SPONSOR (funding authority) Name Galen Kirkhoff Title Vice President, Member Services Office G242 Phone 494-1242	
TYPE OF SERVICE REQUESTED: <input type="checkbox"/> Information Strategy Planning <input checked="" type="checkbox"/> Business Process Analysis and Redesign <input checked="" type="checkbox"/> New Application Development <input type="checkbox"/> Other (please specify) _____ <input type="checkbox"/> Existing Application Enhancement <input type="checkbox"/> Existing Application Maintenance (problem fix) <input type="checkbox"/> Not Sure			
BRIEF STATEMENT OF PROBLEM, OPPORTUNITY, OR DIRECTIVE (attach additional documentation as necessary) The information strategy planning group has targeted member services, marketing, and order fulfillment (inclusive of shipping) for business process redesign and integrated application development. Currently serviced by separate information systems, these areas are not well integrated to maximize efficient order services to our members. The current systems are not adaptable to our rapidly changing products and services. In some cases, separate systems exist for similar products and services. Some of these systems were inherited through mergers that expanded our products and services. There also exist several marketing opportunities to increase our presence to our members. One example includes Internet commerce services. Finally, the automatic identification system being developed for the warehouse must fully interoperate with member services.			
BRIEF STATEMENT OF EXPECTED SOLUTION We envision completely new and streamlined business processes that minimize the response time to member orders for products and services. An order shall not be considered fulfilled until it has been received by the member. The new system should provide for expanded club and member flexibility and adaptability of basic business products and services. We envision a system that extends to the desktop computers of both employees and members, with appropriate shared services provided across the network, consistent with the ISS distributed architecture. This is consistent with strategic plans to retire the AS/400 central computer and replace it with servers.			
ACTION (ISS Office Use Only) <input type="checkbox"/> Feasibility assessment approved <input checked="" type="checkbox"/> Feasibility assessment waived <input type="checkbox"/> Request delayed <input type="checkbox"/> Request rejected Assigned to <u>Sandra Shepherd</u> Approved Budget \$ <u>450,000</u> Start Date <u>ASAP</u> Deadline <u>ASAP</u> Backlogged until date: _____ Reason: _____ Authorized Signatures: <u>Rebecca J. Todd</u> Chair, ISS Executive Steering Body <u>Galen Kirkhoff</u> Project Executive Sponsor			

FORM ISS-100-RFSS (Last revised December, 1999)

图 4-6 一个系统服务请求

- 可见性——一个方案或新系统在多大程度上对客户和/或执行管理层是可见的？可以为这个回答制定一个评级标准。
- 收益——一个方案或新系统大约增加多少年收入或者减少多少年支出。这经常是一种猜测，但如果所有的参与者都参与这个猜测，结果应该是足够保守的。
- 优先权——基于以上的回答，对于每个问题、机会和指示，大家一致同意的优先权是多少？如果预算和进度出了问题，这些优先权将有助于调整项目范围。
- 可能的方案（OPT）——在项目的早期阶段，可能的方案最好以简单的形式表述，例如：a) 不要改变已经使人满意的事物；b) 快速修复；c) 对现有系统的简单到中等规模的改进；d) 重设计现有系统；e) 设计一个新系统。这个任务的参与者十分适合进行这样一种高层次讨论。

问题陈述

项目：会员服务信息系统	项目经理：Sandra Shepherd
创建人：Sandra Shepherd	最后修改人：Robert Martinez
创建日期：2003 年 1 月 9 日	最后修改日期：2003 年 1 月 15 日

问题、机会或指示的简要描述	紧急程度	可见性	年收益（美元）	优先权	建议的方案
1. 按照从收到订单时间开始到向客户发货时间为止度量，平均订单响应时间已经增加到 15 天	ASAP	高	175 000	2	新开发
2. 最近兼并 Private Screenings 公司和 GameScreen 公司，将进一步加大对当前系统的吞吐量压力	6 个月	中等	75 000	2	新开发
3. 目前，有三个不同的订单录入系统分别用于音频、视频和游戏部，每个系统都被设计成与不同的仓库系统接口；所以，将库存合并成为一个单一仓库的企图被延迟	6 个月	中等	515 000	2	新开发
4. 缺少对管理信息和决策支持信息的访问。兼并两个订单处理系统（Private Screenings 公司和 GameScreen 公司的）将使这一点变得更加困难	12 个月	低	15 000	3	在新系统开发后，向用户提供易学易用的报告工具
5. 目前会员和订单文件之间存在数据不一致性	3 个月	高	35 000	1	快速地修改，然后开发新系统
6. Private Screenings 公司和 GameScreen 公司的文件系统与音阶公司的不兼容。业务数据问题包括数据不一致和缺少输入编辑控制	6 个月	中等	不确定	2	新开发。额外的收益可能会增加其紧急程度
7. 有机会向因特网开放订单系统，但安全和控制是需要考虑的问题	12 个月	低	不确定	4	新开发的系统的下一个版本
8. 目前的订单处理系统与正在为仓库开发的自动标识识别（条形码）系统不兼容	3 个月	高	65 000	1	快速地修改，然后开发新系统

图 4-7 问题陈述示例

第 2 章介绍的 PIECES 框架可以用来对问题、机会、指示和约束条件进行分类。例如，图 4-7 中的问题 1 按照 PIECES 可以分类成 P. B. ——性能、响应时间（见图 2-4）。图 4-7 中的问题 4 可以分类成 I. A. 2——信息、输出、缺少所需的信息。

用来完成这个任务的主要技术包括调查研究和同系统所有者开会，这些技术将在第 5 章中讲解。

4.3.2 任务 1.2——协商项目的初步范围

项目范围定义了项目的边界，将被包括进来或者不被包括进来的业务。项目范围在项目执行期间可能会发生变化，但是初始项目计划必须确定项目的初步范围。之后如果项目范围明显地变化了，有关各方将能更好地理解为什么预算和进度也要变化。这个任务可以和前面那个任务并行进行。

通常高级系统分析员或项目经理领导这个任务。大部分其他参与者可以粗略地归为系统所有者，这包括主要负责人、可能受该系统影响的所有组织部门的经理，还可能包括信息系统经理。系统用户、系统设计人员和系统构造人员一般不参与这个任务。

如图 4-5 所示, 这个任务使用了前一个任务产生的初始问题陈述, 问题、机会和指示形成了定义项目范围的基础。项目范围陈述也被添加到资料库中供将来使用, 并作为任务交付成果 (包含项目范围的初始问题陈述) 用文件正式地记录下来。

项目范围可以容易地定义于信息系统构件的上下文中。例如, 项目范围可以按照如下形式描述:

- 什么类型的“数据”描述了正被研究的系统? 例如, 一个销售信息系统可能需要关于“客户”、“订单”、“产品”和“销售代表”之类的数据。
- 正被研究的系统包括什么业务“过程”? 例如, 一个销售信息系统可能包括用于“目录管理”、“客户管理”、“订单输入”、“订单履行”、“订单管理”和“客户关系管理”之类的业务过程。
- 系统如何同用户、地点以及其他系统进行连接? 例如, 一个销售信息系统潜在的接口对象可能包括“客户”、“销售代表”、“销售职员和经理”、“地区销售办公室”以及“应收账款和库存控制信息系统”。

用来完成这个任务的主要技术包括调查研究和会议。许多分析员喜欢组合这个任务及其前后的任务并在一次会议中一并解决。

4.3.3 任务 1.3——评估项目价值

在这个任务中, 我们将回答这样的问题: “这个项目看上去是否值得?” 在项目的早期阶段, 问题的回答可能实际上浓缩成了一种“猜测”——解决问题、探索机会或实现指示能够带来足够的价值以抵消开发这个系统将投入的费用吗? 根据我们目前已经收集到的有限事实, 全面的可行性分析是不可能的。

通常高级系统分析员或项目经理领导这个任务, 但系统所有者 (包括主要负责人、业务部门经理和信息系统经理) 应做出决策。

如图 4-5 所示, 包含项目范围的初始问题陈述启动了 this 任务, 它提供了初步的价值评估所需的信息。这个任务除了“继续或不继续”的决定以外不存在实际的交付成果, 但实际上会有几种不同的决定: 项目可能被批准, 也可能被取消, 项目范围可能要重新协商 (增加或减少)。显然, 只有当项目被认为是值得的并被批准继续时, 范围定义阶段中的其余任务才有必要进行。

4.3.4 任务 1.4——计划项目进度表和预算

如果项目被认为值得继续下去, 我们现在就可以深入地计划项目了。初步的项目计划至少应该包括以下内容:

- 一个初步的主计划, 包括进度和分配给整个项目的资源。这个计划将在项目的每个阶段结束时更新, 它有时被称为基线计划。
- 用于完成项目下一个阶段 (问题分析阶段) 的一个详细计划和进度表。

这个任务是项目经理的责任。大多数项目经理发现让尽可能多的项目团队成员参与进来是有好处的, 这包括系统所有者、用户、设计人员和构造人员。第 3 章使用了联合项目计划来描述制定一个项目计划的团队方法。

如图 4-5 所示, 这个任务由是否继续该项目的决定启动。这个决定代表了一种对于项目范围、问题、机会、指示和价值 (这个“价值”仍需要汇报和批准) 的一致意见。包含项目范围的问题陈述是 (来自资料库的) 关键输入, 而交付成果是基线计划和进度表。工作陈述 (见第 3 章) 和项目进度表以及资源分配也被添加到资料库中供持续的监督和修改。进度表和资源一般在资料库中作为一个项目管理软件文件维护。

第 3 章深入讲解了用来制定项目计划的技术。如今, 项目管理软件 (例如 Microsoft Project) 可以支持这些技术。第 3 章还讨论了完成该计划的详细步骤。

4.3.5 任务 1.5——汇报项目计划

在大多数组织中, 潜在的项目比可用于支持那些项目的人力和财力资源要多得多。除非项目已经被事先确定为具有最高优先权 (通过某种事先的战略规划过程), 否则它就必须向指导部门汇报以获得批准。大多数组织由指导部门批准, 并监视项目及其进展情况。任何指导部门都应该主要由非信息系

统专家或管理人员构成。许多部门指定副总裁主持指导部门工作；另一些组织指定副总裁的直接下属主持指导部门工作；还有一些组织成立了两个指导部门，一个由副总裁主持，一个由副总裁的直接下属主持。信息系统经理服务于指导部门，仅限于回答问题和将优先次序传达给开发人员及项目经理。

理想情况下，主要负责人应同项目经理一起推动这个任务。主要负责人的出席向所有参加动员会的人员直接传达了信任和重视。其他的会议参加人应该包括整个项目团队，这包括系统所有者、用户、分析员、设计人员和构造人员。理想情况下，这个动员会应该对来自企业团体的所有感兴趣的职员开放。这增进了团体对项目的了解和一致意见，同时减少了流言和错误信息。至于内联网站，应该指派一名网站管理员或网站作者参加项目团队。

如图 4-5 所示，这个任务由项目计划的完成触发。问题陈述和项目范围可以从资料库中得到。交付成果是项目章程，通常是一个文档。项目章程包括定义项目的各种要素，其内容包括：参与者，问题、机会和指示，项目范围，方法学，要完成的工作陈述，交付成果，质量标准，进度表以及预算。应该将项目章程添加到项目网站上供所有人查阅。项目章程的要素还可以重新格式化（使用 PowerPoint 之类的软件）为幻灯片和印刷品，以便在项目动员会上使用。

有效的人际沟通能力是这个任务的关键，其中包括：说服、推销变化、商务写作和公共演讲。

范围定义阶段的参与者可能判定该项目不值得投入，指导部门也可能判定其他项目更重要，或者主要负责人可能不再认可这个项目。在这些情况下，项目都要终止，好在只花了很少的时间和精力。另一方面，带着系统所有者和指导委员会的所有祝福，项目现在可以继续进入问题分析阶段。

4.4 问题分析阶段

“除非你已经理解了它，否则不要试图修改它。”这句话恰当地揭示了系统分析的问题分析阶段的任务。实际上，总是会有一个当前的（或者现有的）系统，无论其自动化程度如何。问题分析阶段为分析员提供了对触发该项目的问题、机会和/或指示的更全面的理解。这个阶段要回答问题：“真的值得解决这些问题吗？”和“真的值得构建一个新系统吗？”在其他方法学中，问题分析阶段可能被称做研究阶段、当前系统研究、详细研究阶段或可行性分析阶段。

问题分析阶段很少能够省略，几乎总是需要对当前系统有某种程度的理解，但可能有理由加速问题分析阶段。首先，如果项目是由一个战略或战术规划触发的，项目的价值可能就是毫无疑问的——问题分析阶段将被缩减到仅仅理解当前系统，而不是分析它。其次，一个项目可能是由于一个指示（例如要求符合某个政府指令和最终期限）引发的。在这种情况下，项目的价值也是毫无疑问的。最后，某些方法学和组织有意地合并问题分析阶段和需求分析阶段，从而加速系统分析的过程。

问题分析阶段的目标是充分地研究和理解问题领域并全面分析其中存在的问题、机会和约束条件。某些方法学鼓励对当前系统进行十分深入的理解，并十分详细地使用数据流图之类的系统模型记录当前系统。

图 4-8 是问题分析阶段的任务图，这个阶段最后的交付成果和里程碑是产生处理问题、机会和指示的系统改进目标。根据系统规模的大小（也就是它的复杂度）和对项目价值的已知程度，图 4-8 说明的任务可能需要花上 1~6 个星期，但这些任务中大多数可以通过 JRP 之类的会议加快完成速度。现在让我们更详细地解释每个任务。

4.4.1 任务 2.1——研究问题领域

在问题分析阶段，项目团队首先试图了解当前系统。每个系统所有者、用户和分析员对系统具有不同层次的理解——不同的详细程度、不同的表述方式、不同的感知和不同的观点。组织良好的研究可以对各方都有启迪作用，甚至包括系统自身的管理人员和用户。研究问题领域很重要，业务问题、机会、指示和约束条件都存在于这个领域中。

这个任务将由项目经理领导，但由资深系统分析员主持。一个人同时扮演这两个角色十分常见（正如 Sandra 在音阶公司案例中那样）。其他系统分析员也可以参与进来，进行面谈、为会议做记录和记录调查结果。详细的研究应该包括来自支持该系统和项目或受其影响的所有业务部门的系统所有者

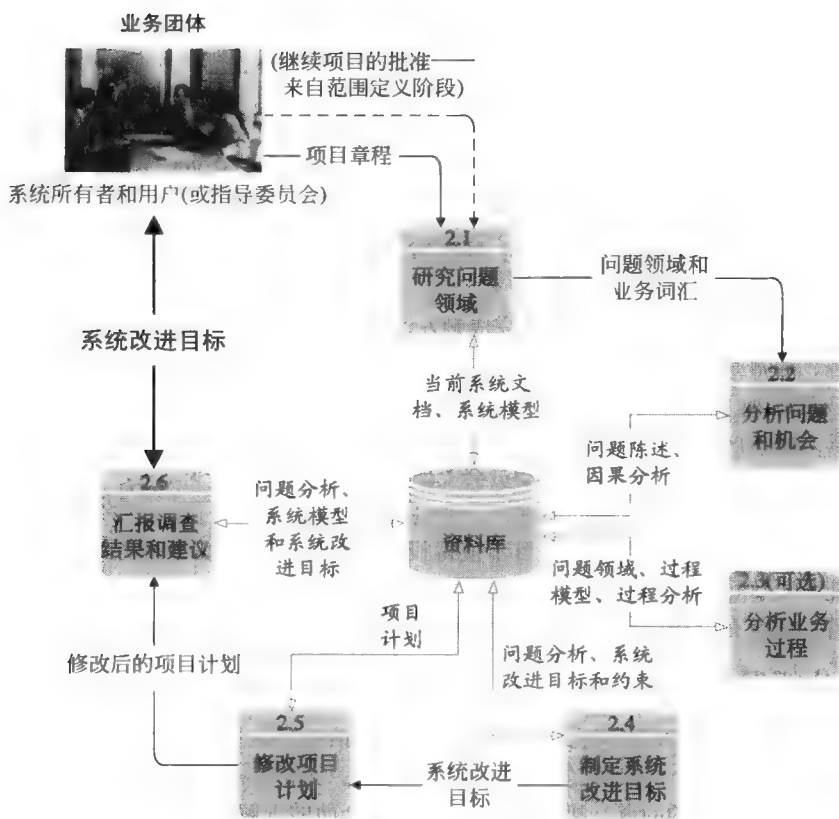


图 4-8 系统分析之问题分析阶段的任务

和用户代表。为了覆盖正被研究的系统的全部范围，包括足够的用户就显得特别重要。在某些组织中，一个或多个有经验的用户被全职“出借”给项目作为业务分析员；但是，很少有哪一个用户能够完全代表所有用户的利益。然而，业务分析员可以作为推动者让合适的人参与进来，同时保持与业务部门和管理层有效的沟通。系统设计人员和构造人员很少参与这个任务，除非他们被请来确定当前系统的某些技术限制。

在图 4-8 中，这个任务由范围定义阶段中对继续该项目的批准所触发（细线表示这个批准是一个事件或触发器，而不是一个数据或信息流）。批准来自系统所有者或指导委员会。主要的信息输入是项目章程和可能存在于当前系统资料库和程序库中的任何系统文档。当前系统的文档并非总是存在，而且即使确实存在，还必须仔细检查其时效性——大部分这样的文档都已过期，因为在一个系统的整个生命周期中，分析员和程序员并不总是用心地修改那些文档。

这个任务的交付成果是对问题领域和业务术语的理解。你对现有问题领域的理解应该以文档形式记录下来，以便验证你是否真的理解了。有多种方式记录问题领域：绘制当前系统的系统模型是其中一种方式，但模型可能导致一种称为“分析麻痹”的现象，也就是创建优秀模型的愿望影响了项目进度；另一种方式是使用信息系统构件作为框架来列出和定义系统领域。

- “知识”——列出所有与系统当前存储的数据（在文件、数据库、表格中等）有关的内容，并按照业务词汇定义每项内容。例如，“一个订单是一个业务事务，其中一位客户请求购买产品。”

另外，我们可以列出现有系统目前生成的所有报告，并描述其用途。例如，“未完成订单报告描述了所有批准供货后一星期内还没有被供应的订单，这份报告被用来通过个人接触启动客户关系管理。”

- “过程”——定义当前为其实现了业务响应（过程）的每个业务事件。例如，“一位客户发出一份新订单”、“一位客户请求修改一份以前发出的订单”或“一位客户取消一份订单”。

- “通信”——定义运行当前系统的所有地点和每个地点的所有用户。例如，“该系统目前用于圣迭哥、达拉斯、圣路易斯、印第安纳波利斯、亚特兰大和曼哈顿的地区销售办公室。每个地区销售办公室有一位销售经理、销售经理助理、行政助理和5~10名销售员，他们都使用当前的系统。每个地区也驻扎了5~30个销售代表，他们大部分时间在路上，但他们每天晚上上载订单和其他事务。”

接口的另一个方面是系统接口——也就是存在于当前信息系统与其他信息系统和计算机应用程序之间的接口。信息系统的职员可以快速列出并描述这些接口。

最后，组织的系统开发方法学和项目计划将决定需要哪种类型和哪种详细程度的文档。

业务术语也经常被遗漏。理解一个系统的业务术语是理解这个系统的一种有效方式，它消除了业务专家和技术专家之间经常存在的或发展出来的沟通障碍。

其他几种技术和技能对理解现有系统也很有用。调查研究技术（在第5章中讲解）对了解任何现有系统都很重要。另外，联合需求计划（JRP技术，也在第5章中讲解）可以加快完成这个任务。最后，与用户清晰地沟通你对系统情况的了解的能力也是至关重要的。

上下文图

上下文图的目的是分析系统如何同它周围的世界交互，并用通常的词汇说明系统的输入和输出。可以用各种方法绘制上下文图。第8章展示了传统的格式，它在绘制数据流图的第1步进行。第6章展示了上下文图的另一种格式。图4-9中的上下文图采用了一种混合格式。它采用了用例符号，因为用例正成为需求分析阶段一般可接受的工具。

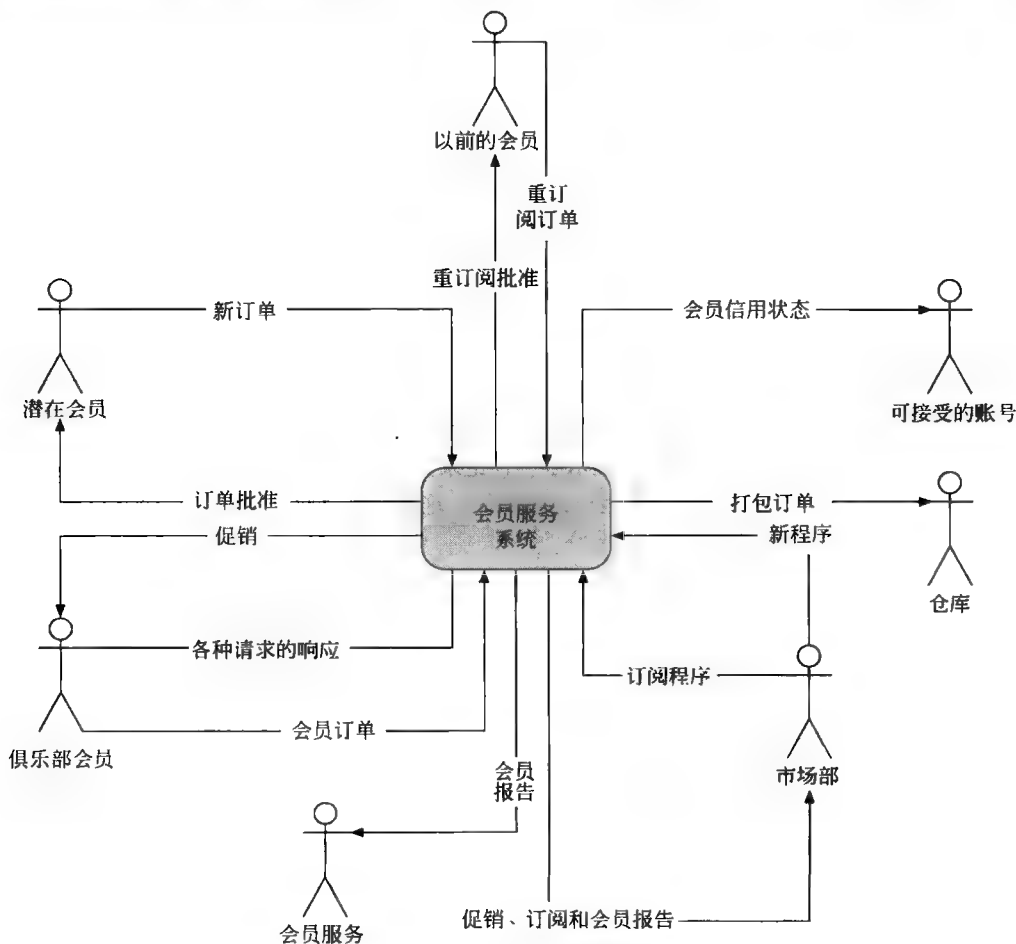


图4-9 上下文图

系统本身被绘制成图形正中的一个“黑盒子”。这里还不准备分析盒子内部。现在，我们只想明白每个人将如何使用这个盒子。图形外围的小人符号代表了将同这个系统交互的人、企业和其他信息系统。在用例图中，它们被称为角色，这里我们也可以这样称呼它们。在传统的数据流图中，它们被称为外部代理。在第6章和第8章中，你将发现一旦我们分析系统盒子的内部，其他事务（例如时间）或设备（例如传感器）也会成为角色或者外部代理。但在上下文图中它们很少出现。

线条表示了角色提供给系统的输入（箭头指向系统）和系统创建的输出（箭头指向角色）。每个输入和输出都用一个描述它的名次词组标识。

要构造一个上下文图，询问用户系统必须响应什么业务事务（它们就是输入）。还要询问用户系统必须生成什么报告、通知和其他输出。系统可能会生成很多报告，这会使图形很快地变得拥挤；将这些报告按需进行归并，以保持图形的可读性。在过程中的其他阶段，这些输出会被分别地分析。

我们当然不能从一个上下文图中构建出信息系统，但它是坚实的第1步。从这幅简单的图中，我们知道了系统必须响应哪些输入，生成哪些输出。换句话说，它帮助我们理解了问题领域。我们将在第6章中看到如何从上下文图中找到用例。那将是打开“黑盒子”的第1步。

4.4.2 任务2.2——分析问题和机会

除了了解当前系统以外，项目团队必须同系统所有者和系统用户一起分析问题和机会。读者可能会问：“问题和机会不是在范围定义阶段的早期就确定了吗？”是的，确实如此，但那些初始问题可能仅仅是其他一些没有被用户很好地了解或理解的问题的症状，况且我们还没有真正地分析那些问题。

真正的问题分析是一项很难掌握的技能，特别是对于没有经验的系统分析员来说更是如此。经验表明，大多数新系统分析员（以及许多系统所有者和用户）没有真正地分析问题就想解决问题。他们可能会这样陈述一个问题：“我们需要……”或者“我们想……”。如果这样做的话，他们就正在以一种方案的形式陈述问题。更老练的问题解决者已经学会了在陈述任何可能的问题之前真正地分析这个问题，他们分析每个问题的原因和结果。结果可能是一个不同的、更深的或更基本的问题的症状，那个问题也必须分析原因和结果，直到原因和结果不再引发其他问题。因果分析可以得出对问题的真正理解，并且能够得出虽然并不十分明显但更有创造性和更有价值的方案。

系统分析员推动这个任务，但是所有的系统所有者和用户应该主动地参与到因果分析中，因为他们是问题领域专家。系统设计人员和构造人员通常不参与这个过程，除非要求他们分析可能存在于当前系统中的技术问题。

如图4-8所示，项目团队对系统领域和业务术语的理解触发了这个任务。这种对问题领域的理解是至关重要的。因为除非他们已经理解了那些问题发生的领域，否则团队就不应该企图分析问题。这个任务的其他信息输入是初始问题陈述（来自范围定义阶段）。这个任务的交付成果是修改的问题陈述以及对每个问题和机会的因果分析。图4-10展示了一种记录因果分析的文档格式。

调查研究技术和JRP技术对这个任务是至关重要的，这些技术以及因果分析都在下一章中讲解。

4.4.3 任务2.3——分析业务过程

这个任务只适用于业务过程重构（BPR）项目、建立在业务过程重构基础上的项目，或者需要重大的业务过程重构的开发项目。在这类项目中，要求项目团队十分详细地检查企业的业务过程，度量每个过程相对于整个组织增加或减少的价值。业务过程分析可能会充满感情色彩，系统所有者和用户可能会极力捍卫他们的现行业务过程，分析员必须保持对事不对人，并经常提醒每个人分析的目标是确定将对企业和企业中的每个人都有利的基本业务变化。

一个或者几个系统分析员或者业务分析员主持这个任务。理想情况下，分析员应该是在BPR方法方面有经验的、受过培训的或通过认证的。其他参与者仅应该包括合适的系统所有者和用户。在业务过程按最大效率重新设计之前，业务过程分析应避免任何专注于信息技术方案的诱惑。有些分析员发现假设存在能够做成任何事情的“完美的人”和“完美的技术”是有用的，他们询问：“如果这个世界是完美的，我们还需要这个过程吗？”

问题、机会、目标和约束矩阵

项目：会员服务信息系统		项目经理：Sandra Shepherd	
创建者：Robert Martinez		最后修改人：Robert Martinez	
创建日期：2003 年 1 月 21 日		最后修改日期：2003 年 1 月 31 日	
因果分析		系统改进目标	
问题或机会	原因和结果	系统目标	系统约束条件
1. 订单响应时间不可接受	1. 吞吐量增加，而处理订单的职员人数减少了。处理一个订单的时间仍保持相对不变 2. 系统过于依赖键盘。对大多数订单来说，都要输入很多相同的数据值。当前系统的结果是每个订单的处理时间比理想情况下要长 3. 数据编辑由 AS/400 进行。由于那台计算机已经达到了工作极限，所以订单编辑响应变慢了。而由于订单处理职员为了更快地处理，结果导致错误增加 4. 订单的仓库提货单的设计没有把订单处理的效率最大化。由于仓库的工作量在增长，所以订单处理延迟是不可避免的	1. 处理一个订单的时间减少 30% 2. 消除所有订单中 50% 的键盘数据录入工作 3. 对于剩余的订单，尽可能地减少键盘输入，可以使用计算机屏幕上的点选对象来代替键盘输入 4. 将数据编辑工作从一台共享的计算机转移到桌面计算机 5. 用会员服务系统和仓库系统之间的无纸化通信代替现有的提货单	1. 不会增加订单处理人手 2. 开发的系统必须与现有的 Windows 95 桌面标准兼容 3. 新系统必须同已经批准的自动识别系统（条形码）兼容

图 4-10 因果分析示例

如图 4-8 所示，业务过程分析任务仅仅依赖于某些问题的领域知识（来自任务 2.1）。这个任务的交付成果是“as is”过程模型和过程分析。过程模型可能非常类似于数据流图（见图 4-2），但它们主要用来显示：1）通过过程的数据量；2）每个过程的响应时间；3）系统中出现的任何延迟或瓶颈。过程分析数据提供了额外的信息，例如：a）每个过程的开销；b）每个过程增加的价值；c）取消或理顺该过程的后果。根据这个“as is”模型和对它们的分析，团队开发出“to be”模型、重设计业务过程、消除冗余和官僚主义，并提高效率和增加服务。

该任务可以使用几种技术。调查研究技术和团队会议（见第 5 章）是很有价值的，过程建模技术（见第 8 章）对 BPR 的成功也是必不可少的。

4.4.4 任务 2.4——制定系统改进目标

给定我们对当前系统的范围、问题和机会的理解，我们现在就可以制定系统改进目标了。这个任务的目的是建立成功的准则，对系统的任何改进都将按照该准则进行度量。这个任务也确定了任何可能限制系统改进的灵活性的约束条件。成功的准则应该按照目标进行度量。目标代表了制定对新系统的预期的首次尝试。除了目标以外，我们还必须确定已知的约束条件。约束条件是针对实现目标的限制或界线。最终期限、预算和所需的技术是约束条件的例子。

系统分析员推动这个任务，其他参与者包括已经参与了问题分析阶段中其他任务的系统所有者和系统用户。再次说明，我们仍然不关心技术问题，所以系统设计人员和构造人员不参与该任务。

这个任务由在任务 2.2 和任务 2.3 中完成的问题分析触发。对于每个验证过的明显问题，分析员和用户应该定义专门的系统改进目标，他们也应该确定可能会限制或阻碍实现这些系统改进目标的约束条件。

系统改进目标应该是精确的、可度量的定义新系统预期的业务性能陈述。图 4-10 最后两列记录了典型的系统改进目标和约束条件。

4.4.5 任务 2.5——修改项目计划

项目范围是不断变化的。以我们在范围定义阶段获得的初步理解和估计为基础，项目范围可能已经在规模和复杂程度上增加或缩减（增加更加常见）。现在我们正在接近问题分析阶段的最后步骤，我们应该重新评估项目范围，并相应地修改项目计划。

项目经理与系统所有者和整个项目团队一起推动这个任务，系统分析员和系统所有者是这个任务中的关键人物。分析员和所有者应该考虑到可能这个新系统并非能够满足所有的目标。新系统可能比预期的规模更大，他们可能不得不减少范围以满足最终期限。在这种情况下，系统所有者将按照重要性排列目标。然后，如果范围必须缩小，则具有较高优先权的目标将告诉分析员什么是最重要的。

如图 4-8 所示，这个任务由系统改进目标的完成触发。初始项目计划是另一个关键输入，修改后的项目计划是关键输出。修改后的计划现在应该包括一个用于后面的需求分析阶段的详细计划。用于修改项目计划的技术和步骤参见第 3 章（项目管理）。

4.4.6 任务 2.6——汇报调查结果和建议

与范围定义阶段一样，问题分析阶段也以沟通任务作为总结。我们必须向业务团体汇报调查结果和建议。项目经理和主要负责人应该一起推动这个任务。其他的与会者应该包括整个项目团队，这包括指定的系统所有者、用户、分析员、设计人员和构造人员。同往常一样，会议应该对来自业务团体的所有感兴趣的人员开放。而且，如果为该项目建立了一个内联网站，在整个问题分析阶段中，网站应该得到不断地维护，确保项目进展的持续沟通。

这个任务由修改后的项目计划的完成所触发。有意义的输入包括问题分析、系统模型、系统改进目标以及在问题分析阶段生成的其他文档。相应的要素组合成系统改进目标。形式可以是一份报告、一次口头汇报或者由审核员或同组人员进行的一次审查（也称为走查）。图 4-11 显示了一份书面报告的提纲。

当前_____系统的分析	
I. 总结（大约 2 页）	IV. 当前系统分析（大约 5 ~ 10 页）
A. 总结建议	A. 性能问题、机会和因果分析
B. 总结问题、机会和指示	B. 信息问题、机会和因果分析
C. 简单陈述系统改进目标	C. 经济问题、机会和因果分析
D. 简单解释报告内容	D. 控制问题、机会和因果分析
II. 背景信息（大约 2 页）	E. 效率问题、机会和因果分析
A. 面谈和小组会议的人员清单	F. 服务问题、机会和因果分析
B. 被探索的其他信息资源清单	V. 详细建议（大约 5 ~ 10 页）
C. 描述使用的分析性技术	A. 系统改进目标和优先权
III. 当前系统概述（大约 5 页）	B. 约束条件
A. 战略意义（如果项目是一个现有信息系统战略规划的一部分或者受到战略规划影响）	C. 项目计划
B. 当前系统的模型	1. 范围重新评估和提炼
1. 接口模型（显示项目范围）	2. 修改后的主计划
2. 数据模型（显示项目范围）	3. 用于定义阶段的详细计划
3. 地理模型（显示项目范围）	VI. 附录
4. 过程模型（只显示功能性分解）	A. 详细的系统模型
	B. 其他相应文档

图 4-11 一份系统改进目标和建议报告的提纲

人际交往能力对这个任务来说是必需的。系统分析员应该能够在不涉及技术问题或技术方案的情况下撰写正式的业务报告并进行业务汇报。

这个任务总结了问题分析阶段。在这个总结阶段之后，必须做出以下决策之一：

- 授权项目继续，也就是说进入需求分析阶段。
- 调整项目的范围、费用和/或进度表，然后继续进行需求分析阶段。

- 取消该项目，由于：1) 缺少资源来进一步开发该系统；2) 意识到问题和机会并不像预期的那样重要；3) 意识到新系统的收益不可能超过开支。

得到系统所有者某种程度的认可后，项目现在就可以继续进行需求分析了。

4.5 需求分析阶段

许多没有经验的分析员在完成问题分析阶段后会犯一个重要错误。他们开始寻找可选方案，特别是技术方案。下面这句话揭示了在新信息系统中一个最常引用的错误，“系统确实可以工作，而且技术上给人留下深刻印象，但就是没有实现我们想要的功能。”需求分析阶段为一个新系统定义业务需求。

上面那句话中的关键是“什么”，而不是“如何”！分析员经常深陷技术方案之中，以至于没有充分地方案定义业务需求。需求分析阶段回答这个问题：“用户需要什么？想从一个新系统中得到什么？”这个阶段是任何一个信息系统成功的关键！在不同的方法学中，需求分析阶段可能被称为定义阶段或者逻辑设计阶段。

能够忽略需求分析阶段吗？决不能！新系统将总是根据它们是否实现了业务目标 and 需求来进行评估，而不管技术方案如何出色或复杂！

需要注意的是：有些方法学将问题分析阶段和需求分析阶段合并为一个阶段。

图 4-12 展示了需求分析阶段的典型任务。最后的交付成果和里程碑是产生一个“业务需求陈述”，它将实现在前面阶段中确定的系统改进目标。在任务图中你可能注意到的首要事情之一是大部分任务不像在以前的任务图中那样是顺序的。相反，随着团队工作逐步完成需求陈述，这些任务中有许多可以并行进行。现在让我们更详细地介绍这些任务。

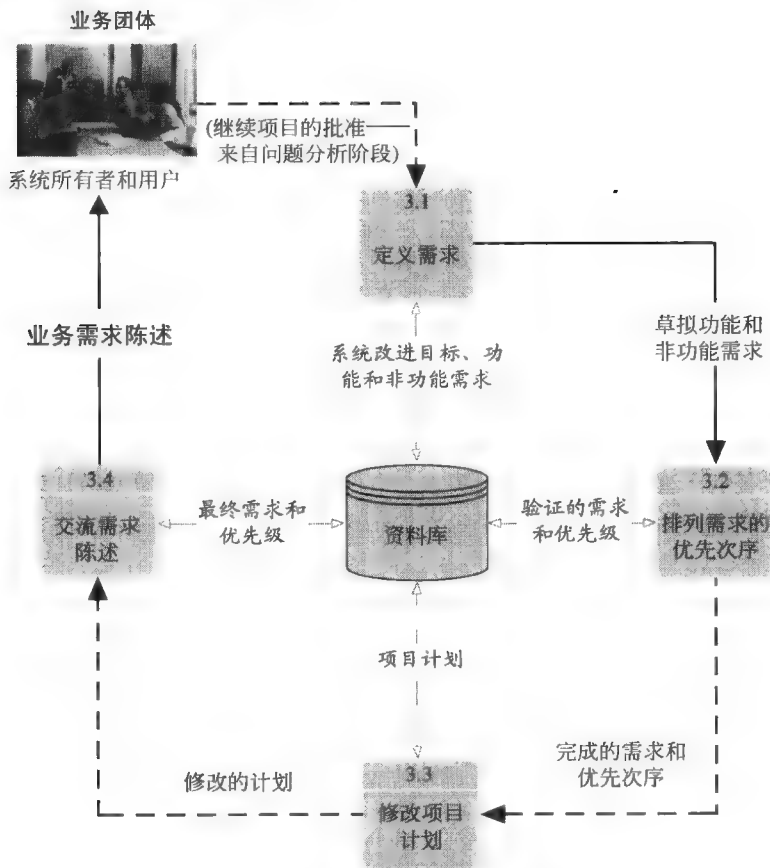


图 4-12 系统分析之需求分析阶段的任务

4.5.1 任务 3.1——定义需求

需求分析阶段的第一个任务是确定需求。虽然这看上去是一项容易或者微不足道的任务，但它常常是许多错误、忽略和冲突的来源。这个任务的基础建立在我们确定系统改进目标时的问题分析阶段中，它至少要将那些目标转换成满足其需要的功能需求和非功能需求的框架。功能需求经常以满足系统改进目标所需的输入、输出、过程和存储的数据的形式定义。非功能需求的例子如：性能（吞吐量和响应时间），易学易用性，预算、开支和开支节省，时间表和最终期限，文档和培训需求，质量管理，安全和内部审核控制。

系统分析员推动这个任务，他们也对结果做文档记录。显然，系统用户是业务需求的主要来源。某些系统所有者也可能会参与到这个任务中，因为他们在制定指导这个任务的系统改进目标中扮演了某个角色。系统设计人员和构造人员不应该参加这个任务，因为他们往往将注意力过早地导向技术问题和技术方案。

如图 4-12 所示，来自问题分析阶段的对继续该项目的批准触发了这个任务（和阶段）。关键的输入是来自问题分析阶段的系统改进目标（通过资料库）。当然，当需要时来自问题分析阶段的所有相关信息都可以从资料库中获得以供参考。

这个任务唯一的交付成果是功能需求和非功能需求的草稿，各种形式都可以。在最简单的形式中，框架可以分成 4 个逻辑部分系统改进目标的原始清单和对每个目标的一个子清单，其内容包括实现这个目标需要的：a) 输入；b) 过程；c) 输出；d) 存储的数据。但越来越多的系统分析员采用称为用例的建模工具表述业务需求。用例对一个新系统必须处理的业务场景和事件进行建模。在第 6 章将介绍用例，它应用于全书。

前面用来确定问题、机会和约束条件的 PIECES 框架也可以用作定义需求草案的框架。

有几种技术可以用在这个任务中。联合需求计划（JRP）是用于快速整理业务需求的首选技术。另外，分析员也可以使用其他调查研究方法，例如调查和面谈。JRP 技术和调查研究技术都将在下一章中讲解。

4.5.2 任务 3.2——排列需求的优先次序

早先我们提到一个系统开发项目的成功可以按照业务需求被满足的程度来度量，但不是所有的需求都应该被平等对待。如果一个项目落后于进度或者超出预算，知道哪个需求比其他需求更重要可能是很有用的。因此，给定验证过的需求后，系统所有者和用户应该排列系统需求的优先次序。

排列需求的优先次序可以使用一种称为时间盒的技术。时间盒技术试图将需求分割成“段”，这些需求可以在一段时间内实现，而不会超出用户和管理层的耐心。时间盒技术要求我们理清需求的优先次序。

系统分析员推动这个优先权划分任务，系统所有者和用户制定实际的优先权，系统设计人员和构造人员不参与这个任务。该任务由被验证的需求、事件或里程碑触发。要知道，你不可能恰当地给一组不完整的需求排序。这个任务的交付成果包含优先权的需求，优先权可以按照需求的相对重要性进行分类。

- 强制的需求是最小的系统（版本 1.0）必须实现的需求。没有它们系统就是没用的。注意，我们往往会把太多的需求标记为强制的需求。一个强制的需求不能再分级，因为它对任何方案来说都是基本需求。事实上，如果一个强制需求可以被分级，那么它实际上应该是一个理想的需求。
- 理想的需求对版本 1.0 来说是非基本的需求，但它可能对某些未来的版本仍是基本的。理想的需求可以并且应该被分级，使用版本号作为分级策略是交流和分类理想需求的一种有效方式。

4.5.3 任务 3.3——修改项目计划

现在我们已经确定了系统业务需求，我们将回来重新定义对项目范围的理解并相应地修改项目计划。项目团队必须考虑到新系统可能会比原先预期的规模更大。如果是这样，项目团队就必须相应地

调整进度、预算或范围。我们还应该确保批准使该项目继续进入下一个阶段（设计阶段的工作可能已经开始了，但是决策仍需要评审）。

项目经理连同系统所有者和整个项目团队一起推动这个任务。通常，项目经理和系统所有者是这个任务中的关键人物。他们应该考虑到需求现在可能超出了为项目和新系统建立的最初构想。他们可能不得不削减范围以满足最后期限或者增加预算来完成工作。

如图 4-12 所示，被批准的需求（包括优先权）的完成触发了这个任务。最新的项目计划是另一个关键输入，在资料库中也对它做了相应的修改。用于项目计划维护的工具、技术和步骤已在第 3 章中介绍过。

4.5.4 任务 3.4——交流需求陈述

交流是需求分析阶段一个持续的任务。我们必须在整个阶段都保持同业务团体交流需求及其优先级。用户和管理层经常游说他们对需求和优先级的考虑。交流是不同的观点进行融合的过程。项目经理和主管应该主持这个任务。如今，项目内联网站或门户常用来交流需求。有些系统允许用户和管理人员订阅需求文档，以便确保当需求发生变化时及时得到通知。人际关系、交流和协商技巧是该任务需要的基本技术。

4.5.5 持续不断的需求管理

需求分析阶段现在结束了，也许还没有？在开始系统设计和构造阶段之前冻结业务需求曾经很流行。但如今的经济步伐变得越来越快，企业需要具有快速适应不断变化的需求和机会的能力。信息系统不能比企业自身的响应还慢，因此需求分析不会真正地结束。虽然我们平静地转移到项目的剩余阶段，但在项目开发过程中和系统的生命周期中，我们仍需要连续地管理需求。

需求管理为系统所有者、用户、分析员、设计人员和构造人员定义了一个过程，该过程用于对系统需求变化的建议的提交。这个过程规定了如何请求和记录需求变化，如何登记和跟踪需求变化，什么时候和如何评估需求变化优先权，以及如何最终实现需求变化（如果将被实现的话）。

4.6 逻辑设计阶段

并非所有的项目都采用模型驱动的开发方法，但大多数项目都包括一定的系统建模。逻辑设计阶段使用系统模型进一步记录业务需求，这些系统模型表示了数据结构、业务过程、数据流和用户接口（越来越多的模型采用对象模型，如本章前面介绍的）。从某种意义上说，逻辑设计验证了前面阶段建立的需求。

图 4-13 显示了逻辑设计阶段的主要任务。最后的交付成果和里程碑是生成一个业务需求陈述，它将实现在前面阶段中确定的系统改进目标。在任务图中你可能注意到的首要事情之一是大部分任务不像在以前的任务图中那样是顺序的。相反，随着团队工作逐步完成需求陈述，这些任务中有许多可以并行进行。下面让我们详细地介绍每个任务。

4.6.1 任务 4.1a——结构化功能需求

逻辑设计的一种方法是结构化功能需求。使用敏捷方法，这意味着你应该绘制一个或多个系统模型以演示功能需求。这些模型可能包括那些实际描述了业务需求和用户需求（但不含任何技术方案）的数据、过程和对象模型的组合。除非所有的功能需求都已经建模了，系统模型不会是完整的。通常模型辅以详细的逻辑说明，描述数据属性、业务规则和策略等。

系统分析员主持这个任务，他们也记录结果。显然，系统用户是绘制模型所需的细节信息的主要来源。如图 4-13 所示，该任务由每个功能需求触发。输出是实际的系统模型和详细的规格说明。详细的程度依赖于采用的方法学。敏捷方法通常要求“正好足够”的文档。多少是足够呢？这个问题很难回答。但敏捷方法要求每个交付成果都是以后设计和编程阶段必需的。本书将讲授各种用于逻辑设计的系统建模工具和技术。

4.6.2 任务 4.1b——建立功能需求的原型（可选）

原型是系统建模的替代方法（有时是前提条件）。有时用户难以表达绘制足够的系统模型所需的事

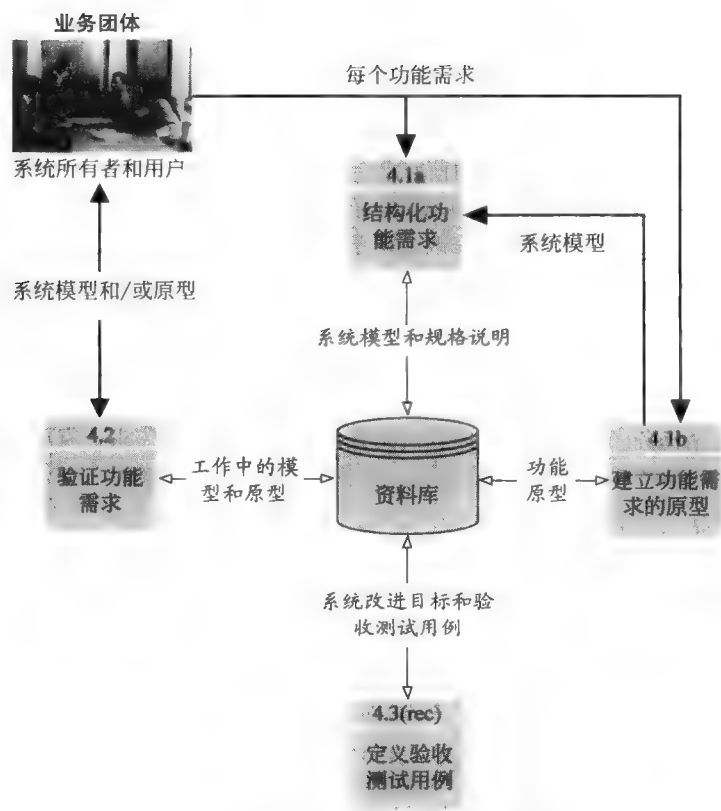


图 4-13 系统分析之逻辑设计阶段的任务

实。这时，一个替代或者补充的方法就是构建原型。原型一般用在需求分析阶段构造输入输出的示例。这些输入输出有助于构造底层的数据库，以及向数据库输入数据和从数据库输出数据的程序。尽管原型是可选的，但它经常用于系统开发项目中，特别是用户难以陈述或者可视化他们的业务需求时。其中基本假设的是：当用户看到时，将能够知道他们的需求。

系统构造人员推动这个任务。系统分析员记录和分析结构。通常，系统用户是任务的主要数据输入者。图 4-13 说明了这个任务依赖于用户已经确定的一个或多个功能需求。系统构造人员和分析员通过构造原型做出响应。如本章前面所述，可以从原型数据库和程序库直接逆向工程出一些系统模型。

4.6.3 任务 4.2——验证功能需求

系统模型和原型是用户需求的表示，必须验证其完整性和正确性。系统分析员主持该任务，协调用户找出需求中的错误、忽略并进行澄清。

4.6.4 任务 4.3——定义验收测试用例

尽管该任务不是必需的任务，但大多数专家认为现在开始规划系统测试并不算早。系统模型和原型有效地定义了新系统的过程需求、数据规则和业务规则。相应地，这些规则说明可以用来定义最终用来测试程序正确性的测试用例。系统分析员或系统构造人员执行该任务，并与系统用户一起验证测试用例。

回顾项目早期定义的系统改进目标，测试用例也可以测试这些目标。

4.7 决策分析阶段

给定一个改进的信息系统的业务需求后，最后我们可以涉及新系统（包括基于计算机的替代方案）如何实现的问题。决策分析阶段确定候选方案，分析那些候选方案并推荐一个将被设计、构造和实现

的目标系统。有时某些人可能已经拥护某个技术方案，但替代的方案（也许是更好的方案）几乎总是存在的。在决策分析阶段，你有必要确定各种可选的方案，分析它们，然后根据分析结果推荐最佳方案。

图4-14列出了决策分析阶段中的典型任务。最后的交付成果和里程碑是产生一个能够实现前面各阶段确定的业务需求的系统方案建议。现在让我们更详细地介绍这些任务。

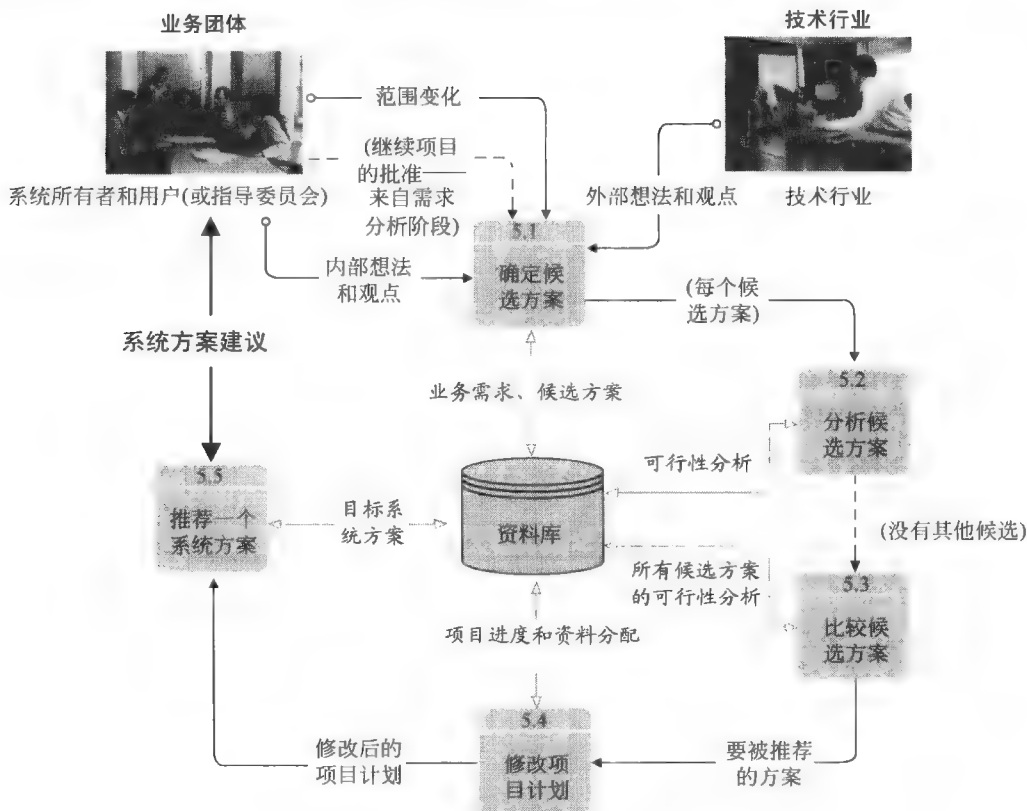


图4-14 系统分析之决策分析阶段的任务

4.7.1 任务5.1——确定候选方案

给定系统分析的定义阶段确定的业务需求后，我们首先必须确定候选方案。某些候选方案是由系统所有者和用户的设计思想和观点形成的，另一些候选方案可能有各种来源，包括：系统分析员、系统设计人员、技术顾问和其他信息系统专家，并且某些技术选择可能受到一个预先定义的并已被批准的技术架构的限制。评估候选方案不是这个任务的目的，该任务仅仅定义要被考虑的可能的候选方案。

系统分析员推动这个任务。系统所有者和用户一般不直接参与这个任务，但他们可能对启动这个任务提出思想和观点。例如，一位系统所有者或用户可能阅读过一篇文章，或者也许知道某些竞争者或熟人的类似系统是如何实现的。无论如何，考虑这些想法都是明智的。系统设计人员和构造人员（例如数据库管理员、网络管理员、技术架构师和程序员）也都是思想和观点的源泉。

如图4-14所示，由于需求分析阶段批准了项目的继续进行，从而正式触发了该任务。在现实情况中，从范围定义阶段开始就应该产生和收集各种想法和观点——通过一些问题解决的过程给出建议方案是人类的天性。除了项目团队以外，内部或外部资源都可能会产生出思想和观点。每个产生的想法都应被考虑为业务需求的一个候选方案。

描述任何一个候选方案的特征的信息量可能会变得很大。候选矩阵（见图4-15）是一种有效地收集、组织和比较不同候选方案特征的有用工具。

特 征	候选方案 1	候选方案 2	候选方案 3	候选方案……
计算机处理的部分 简单描述在这个候选方案中将被计算机处理的部分	将购买 Entertainment Software Solutions 公司的 COTS 软件包 Platinum Plus 并对它进行定制以满足会员服务系统需要的功能	会员服务系统以及与订单履行有关的仓库职能	同候选方案 2	
收益 简单描述这个候选方案将实现的业务收益	可以快速实现这个方案，因为它是一个购买的方案	对用户所需的音阶公司的业务过程给予充分支持，还要与会员账号更有效地交互	同候选方案 2	
服务器和工作站 描述支持这个候选方案需要的服务器和工作站	技术上架构要求使用 Pentium Pro、MS Windows NT 类服务器和 Pentium、MS Windows NT 4.0 工作站（客户端）	同候选方案 1	同候选方案 1	
需要的软件工具 设计和建造这个候选方案需要的软件工具（例如，数据库管理系统、仿真程序、操作系统、语言等）。如果购买应用软件包，则这一条通常没有意义	用于定制软件包的 MS Visual C++ 和 MS Access，用来提供报告编写和集成功能	MS Visual Basic 5.0、System Architect 3.1、Internet Explorer	MS Visual Basic 5.0、System Architect 3.1、Internet Explorer	
应用软件 描述要购买、构建、评估的软件	软件包方案	定制方案	同候选方案 2	
数据处理方法 通常是以下的某些组合：联机、批处理、延迟批处理、远程批处理和实时	客户/服务器	同候选方案 1	同候选方案 1	
输出设备和含义 描述要使用的输出设备、特殊的输出需求（例如，网络、预打印的表格等）以及输出因素（例如，定时约束）	(2) HP4MV 部门级激光打印机 (2) HP5SI LAN 激光打印机	(2) HP4MV 部门级激光打印机 (2) HP5SI LAN 激光打印机 (1) PRINTRONIX 条形码打印机（包括软件和驱动程序）必须设计适应 VGA 分辨率的 Web 页面。所有的内部屏幕将按照 SVGA 分辨率设计	同候选方案 2	
输入设备和含义 描述要使用的输入方法、输入设备（例如，键盘、鼠标等）、特殊输入需求（例如，输入数据的新表单或者改进的表单）以及输入因素（例如，实际输入的定时要求）	键盘和鼠标	Apple “Quick Take” 数码相机和软件 (15) PSC Quickscan 激光条形码扫描仪 (1) HP Scanjet 4C 平板扫描仪 键盘和鼠标	同候选方案 2	

图 4-15 一个候选系统矩阵

特 征	候选方案 1	候选方案 2	候选方案 3	候选方案……
存储设备和含义 简单描述将存储什么数据, 从现有存储访问什么数据, 使用什么存储介质, 将需要多少存储空间, 以及如何组织数据	MS SQL Server DBMS, 100GB 阵列存储功能	同候选方案 1	同候选方案 1	

图 4-15 (续)

调查研究技术和小组会议技术(例如 JRP)是用于研究候选系统方案的主要技术, 这些技术将在下一章中介绍。另外, 第 10 章(可行性分析和系统方案建议)将讲授如何生成候选系统方案并在矩阵中记录这些方案。

4.7.2 任务 5.2——分析候选方案

每个候选系统方案都必须进行可行性分析。当每个候选方案被确定时或者所有的候选方案都已经确定后, 我们都可以进行这项分析。可行性分析不应该在分析费用和收益上受到限制, 大多数系统分析员按照至少 4 条准则来评估各个方案:

- 技术可行性——该方案技术上是否能够现实? 职员具有设计和构造这个方案的技术专业知识吗?
- 运行可行性——该方案实现了用户的需求吗? 实现到什么程度? 这个方案会给用户的工作环境带来哪些改变? 用户对这样一个方案的感觉如何?
- 经济可行性——该方案划算吗?
- 进度可行性——该方案可以在一个可接受的时间段内设计和实现出来吗?

当完成这项任务时, 分析员和用户必须小心, 不要去比较这些候选方案。对每个候选方案单独进行可行性分析而不考虑其他候选方案的可行性。这种方法不鼓励分析员和用户过早地做出关于哪个候选方案是最优的决策。

系统分析员推动这个任务。通常由系统所有者和用户分析运行可行性、经济可行性和进度可行性。通常系统设计人员和构造人员对那些分析做出贡献并在分析技术可行性中扮演重要角色。

图 4-14 显示了每个候选方案的完成都引发这个任务, 但是, 推迟任务到所有的候选方案都确定下来再进行分析也是可以接受的。实际的可行性分析的输入来自各种团队成员, 但是外部专家(和影响力)也经常为此提供数据。每个候选方案的可行性分析都存储在资料库中供日后与其他候选方案进行比较之用。

调查研究技术再次在系统分析任务中起作用, 但对候选系统方案的可行性分析的实施能力是最基本的, 这个技术在第 10 章中讲解。

4.7.3 任务 5.3——比较候选方案

一旦完成了对每个候选方案的可行性分析后, 现在就可以比较这些候选方案, 从中选出一个或多个方案推荐给系统所有者和用户。首先, 任何不可行的候选方案通常从进一步考虑中排除掉。既然是在寻找那些剩下方案中最可行的方案, 所以就要确定和推荐在技术可行性、运行可行性、经济可行性和进度可行性方面提供“整体最优”组合的候选方案, 不过一个候选方案很少会在所有准则上都达到最优。

系统分析员推动这个任务, 系统设计人员和构造人员应该参与回答技术可行性问题。但最终, 应该授权系统所有者和用户执行最后的分析和推荐工作。

在图 4-14 中, 这个任务由所有候选方案(不再有更多的候选方案)的可行性分析的完成所触发, 其输入是所有候选方案的可行性分析结果。可以使用一个矩阵来沟通有关候选方案的大量信息, 图 4-16 中的可行性矩阵可以对候选方案的各种可行性分析进行逐条比较。

这个任务的交付成果是要被推荐的方案。如果推荐不止一个方案, 则应该设置优先权。

可行性分析技术(和可行性分析矩阵)将在第 10 章中讲解。

可行性准则	权 重	候选方案 1	候选方案 2	候选方案 3	候选方案……
运行可行性 功能： 描述候选方案将给组织带来多大的收益以及系统将工作得多好 政策： 描述从用户管理层、用户和组织观点来看这个方案的被接受程度	30%	仅支持会员服务部的需求，而且当前的业务过程将不得不被修改以发挥软件功能优势 得分：60	完全支持用户需要的功能 得分：100	同候选方案 2 得分：100	
技术可行性 技术： 评估支持这个候选方案所需的计算机技术的成熟度、可用性（或获取能力）和必要性 专业知识： 评估开发、运行和维护这个候选系统所需的技术专业知识	30%	Platinum Plus 软件包的当前发行版是版本 1.0，并刚上市 6 个星期。产品的成熟度是一个风险，而且公司需要为技术支持支付额外的月租 需要雇用或培训 C++ 专业人员进行修改以实现集成需求 得分：50	尽管当前的技术人员仅有 PowerBuilder 经验，但是看过 MS Visual Basic 演示和汇报的高级分析员一致认为转换将是简单的，而且招聘到有经验的 VB 程序员要比招聘到 PowerBuilder 程序员更容易且费用更低 从版本号来看，MS Visual Basic 5.0 是一个成熟的技术 得分：95	尽管当前的技术人员熟悉 PowerBuilder，但管理层对 Sybase 公司的 PowerBuilder 的前途感到担心。MS SQL Server 是目前公司的技术标准，并在客户/服务器 DBMS 市场上同 Sybase 竞争。因此，对 PowerBuilder 的未来版本将与我们目前版本的 SQL Server 一同工作时应能表现良好没有保证 得分：60	
经济可行性 开发费用： 回收期（贴现的）： 净现值： 详细计算：	30%	大约 350 000 美元 大约 4.5 年 大约 210 000 美元 见附件 A 得分：60	大约 418 040 美元 大约 3.5 年 大约 306 748 美元 见附件 A 得分：85	大约 400 000 美元 大约 3.3 年 大约 325 500 美元 见附件 A 得分：90	
进度可行性 评估这个方案将花多长时间进行设计和实现	10%	少于 3 个月 得分：95	9 ~ 12 个月 得分：80	9 个月 得分：85	
评分	100%	60.5	92	83.5	

图 4-16 一个可行性分析矩阵

4.7.4 任务 5.4——修改项目计划

读者可能已经注意到有一个不断重复出现的主题始终贯穿本章——当一步步地了解了系统、问题、需求和方案时，我们不断地修改项目计划并相应地调整项目范围。因此，根据我们推荐的方案，应该再一次重新评估项目范围，并相应地修改项目计划。

项目经理及系统所有者和整个项目团队一起推动这个任务，系统分析员和系统所有者在这个任务中是关键人物。但由于我们正在转变到技术性系统设计，所以需要在项目计划修改中包括系统设计人员和构造人员。

如图 4-14 所示,被推荐的方案的完成触发了这个任务。最新的项目进度和资源分配必须被重新评审和修改。修改后的项目计划是主要的输出,它现在应该包括一个用于系统设计阶段的详细计划。用于修改项目计划的技术和步骤在第 3 章(项目管理)中已讲解过。

4.7.5 任务 5.5——推荐一种系统方案

同范围定义阶段和问题分析阶段一样,决策分析阶段也以一个沟通任务结束。我们必须向业务团体推荐一种系统方案。

项目经理和主要负责人应该联合推动这个任务。其他的与会者应包括整个项目团队,其中包括系统所有者、用户、分析员、设计人员和构造人员。通常,这个会议应该对来自业务团体的所有职员开放。如果为该项目建立了一个内联网站,那么在整个问题分析阶段该网站应该被不断地维护,以确保项目进展的连续沟通。

修改后的项目计划的完成触发该任务。目标系统方案(来自任务 4.3)被重新格式化,用于作为系统建议的演示报告,形式可以是一份报告、一次口头汇报或者由一位审核员或同组人员进行的审查(称为走查)。图 4-17 显示了一个书面报告的提纲。

I. 引言	II. 使用的工具和技术	V. 建议
A. 报告的目的	A. 生成的方案	VI. 附录
B. 产生这个报告的项目背景	B. 可行性分析(成本效益分析)	
C. 项目范围	III. 信息系统需求	
D. 报告的结构	IV. 替代方案和可行性分析	

图 4-17 一个典型的系统建议的提纲

人际交流能力对这个任务来说是基本技能,软技能(例如推销和说服)也很重要,许多学校都提供关于这些主题的演讲和沟通方面的课程。系统分析员应该能够撰写正式的业务报告并能够做不涉及技术问题或技术方案的业务演示汇报。

这个任务结束了决策分析阶段,而且也结束了系统分析过程。

复习题

- 什么是驱动系统分析的业务因素?基于这些因素,系统分析应该涉及什么内容?
- 什么是模型驱动分析?为什么使用它?举几个例子说明。
- 结构化分析主要关注什么内容?
- 信息工程主要关注什么内容?
- 面向对象分析为什么流行?它解决了什么问题?
- 系统分析的 5 个阶段是什么?
- 范围定义阶段的目标是什么?
- 在范围定义阶段要执行的 5 个任务是什么?
- 什么是交流项目计划的触发器,谁是观众?为什么交流项目计划很重要?
- 为什么许多新的系统分析员不能有效地分析问题?他们如何才能更有效呢?
- 用于确定和表示系统的功能需求的流行工具是什么?
- 用于排列系统需求优先级的常用工具是什么?
- 可以使用何种原型代替系统建模来决定功能需求?
- 为什么需要决策分析阶段?
- 确定候选方法有哪些方法?

问题和练习

- 存在许多不同的系统分析方法。尽管这些方法不同,广泛接受的系统分析的定义是什么?通常认为系统分析何时开始何时结束?作为一名项目经理,关于系统分析的定义需要知道哪些重要内容,在你的企业中什么是重要的?
- 作为一名系统分析员,在工作中你将面对并使用许多不同的方法。理解每种方法的概念、它们的基本差别、优点和缺点很重要。考虑结构化分析、信息工程和数据建模,以及面向对象分析的区别,它们都代表了模型驱动的分析,并填写下表。

	中心点（数据、过程等）	使用的模型的类型	基本差别
结构化分析			
信息工程和数据建模			
面向对象分析			

3. 增量式系统分析方法基于这样的前提：原型有助于比其他方法更快地展示最重要的业务需求。描述两种最常用的增量式系统分析方法。它们完成哪些工作？如何完成？你对原型法有什么批评意见？增量式系统分析方法可以完全代替更形式化的方法吗，例如结构化分析？

4. 在范围定义阶段，你决不应该漏掉的第一个问题是什么？你如何回答这个问题？在范围定义阶段应该有哪5个任务？

5. 你是一名新系统分析员，你渴望在你的第一个项目中证实自己的能力。在一个与系统所有者和用户召开的问题分析会议上，当你说到“我们需要做这些来解决这个问题”时，你会落入什么常见的陷阱的危险中？你应该使用什么技术避开这个陷阱？

6. 你的项目团队已经完成了范围定义阶段，目前正在问题分析阶段以确立系统改进目标。作为项目团队中的一名系统分析员，你是定义系统改进目标的脑力激荡会议的主持人。由于几个项目所有者和用户以前都没有做过这件事，描述好的系统改进目标的特征，并举几个例子。项目团队的成员建议了以下目标：

a. 降低处理订单所需的时间。

b. 新的系统必须使用 Oracle 存储数据。

c. 数据输入屏幕必须重新设计以使它们更友好。

d. 联机订单处理的顾客满意率必须增加 10%。

这些是好的系统改进目标吗？为什么是，或者为什么不是？如果不是，应该如何重新表述？另外，目标经常具有联系在一起的约束；如果有，你认为与这些目标相关的约束是什么？

7. 你们已经完成了项目的问题分析阶段，现在正在开始需求分析阶段。在关于业务需求的第1次会议上，项目团队中的其他一名分析员询问系统用户，“新系统应该如何满足你们的需求？”这个分析员犯了什么错误？犯这种错误通常的后果是
- 什么？

8. 功能需求和非功能需求的差别是什么？将它们如此分类的目的是什么？分析员可以记录功能性系统需求的两种格式是什么？

9. 将系统需求排列优先级重要吗？需求应该在什么时候排列优先级？可以使用哪种技术？强制性需求和期望需求的差别是什么？什么方法可以用来测试强制性需求是否真是强制性的？

10. 一旦确定系统需求并排列优先级，是否每件事都不用固定下来以便防止范围蔓延或特征蔓延？不需要更新项目计划或者允许关联人员继续要求变更而延迟了系统设计和构造吗，也许甚至延迟项目完成？

11. 为什么接受性测试用例应该在逻辑设计阶段定义？毕竟，技术设计还没有进行，更不用说构建系统了。测试活动不是应该直到构造活动实际开始时还在进行吗？

12. 逻辑设计阶段与需求分析阶段有什么不同？

13. 假设你属于一个项目团队，项目在需求分析阶段遇到了大量困难，并且比进度落后了几个星期。项目经理想通过跳过或者省略逻辑设计阶段的一些任务来试图赶上进度。项目经理认为，现在大家对需求已经有了清楚的认识，实际上设计人员和构造人员经验丰富，为了进行技术设计他们并不真正需要逻辑设计。为了赶上进度，这是合理的方法吗？可能会有什么后果？

14. 在确定和定义了可能的候选方案后，项目中涉及到的各种关联人员的典型角色是什么？

15. 你是一名系统分析员，被要求主持几个候选系统方案的可行性分析和评估。你一般会使用哪套评估标准？在这个任务中你会涉及到谁？你应该在这时将候选方案进行相互比较吗？为什么，或者为什么不？这个任务产生的典型发布物是什么？

项目和研究

1. 选择一个你熟悉的信息系统，基于你作为雇员、客户、其他系统用户或系统所有者的经验，你觉得哪个需要改进？根据需要交换的角色和视角，
- 完成以下工作：

a. 描述你选择的信息系统的特征。

b. 描述拥有和维护信息系统的企业。

- c. 确定初步的问题和机会，任务 1.1。
 - d. 开发一个初始问题陈述，使用图 4-7 描述的格式。
2. 假设你现在是问题 1 中描述的项目的系统分析员。执行管理层对你在问题陈述方面的工作成绩很满意。因此，他们已经让项目继续进行，计划项目进度和预算已经完成，项目计划已经被执行管理委员会批准。作为一名系统分析员，你现在被要求做以下工作：
- a. 开发并记录你对问题领域和业务词汇的理解，使用书中任务 2.1 描述的信息系统构件框架。
 - b. 使用因果分析（任务 2.2）来分析问题和机会。
 - c. 分析业务过程并开发过程模型（任务 2.3）。
 - d. 建立系统改进目标（任务 2.4）。
 - e. 以图 4-10 为例，准备一个问题、机会、目标和约束矩阵。
3. 交流发现和建议是问题分析阶段的最后一个任务。作为项目中的一名系统分析员，要求你准备系统改进目标和建议报告。在本练习中，只准备报告的总结部分，使用图 4-11 显示的格式。执行委员会将使用这个总结对建议做出决定。
4. 到目前为止你在项目中的突出表现继续给管理层留下深刻印象。你的待遇提高了，并要求你继续

进行需求分析阶段。分别是：

- a. 确定系统需求，准备一份功能需求和非功能需求大纲，任务 3.1。既然你的企业使用结构化分析，并且没有使用用例建模，请列出每个系统改进目标以及已经满足每个目标所需的输入、过程、输出和存储的数据。
 - b. 假设在前面步骤中你确定的需求已经被验证。按照它们的相对重要程度排列优先级，使用任务 3.2 中描述的方法。
5. 你的工作使得项目大大地提前于进度，所以执行管理层给你几个星期的带薪休假。当你回来时，项目进入了决策分析阶段。你的下一个任务是确定候选方案。
- a. 描述确定候选方案的过程。在这时你应该小心不要做什么？
 - b. 开发一个候选系统矩阵，以图 4-15 的格式为例，该矩阵包括 3 个可能的方案。
6. 确定了候选方案后，下一步就是分析这些方案。
- a. 描述分析候选方案的过程。在完成这个任务时项目团队不应该做什么？
 - b. 开发一个可行性分析矩阵，基于前面问题中确定的候选方案，使用图 4-16 展示的格式。决定你应采用的权重因子。

小型案例

- 1. 你是一个大型零售店的 CIO。最近，你阅读了 2004 年 12 月 21 日的《华尔街杂志》上的一篇文章“销售地板上的间谍 XXX”。你发现你的竞争对手正在使用视频挖掘来分析客户行为。你的公司也应该采用这个工具（视频挖掘）吗？你的公司对你的竞争者的行动采取什么策略？创造了什么机会？还是威胁？
- 2. 阅读 *Sloan Management Review* Summer 1995 上 Cooper 和 Markus 写的“人类再工程”。在这篇文章中，Okuno 对改变持一种积极的态度。他是如何做到的？讨论为了技术实现的成功，雇员接受改变的重要性？
- 3. 对于小型案例 1。作为 CIO，你相信在你的零售店实现视频挖掘带来的业务收益将超过任何负面的客户感觉。你的公司是儿童的 R U，玩具 R U 的一个子公司。为这个投资作一个经济可行性研究。确保包括了对无形成本和收益的清单，以及你选择的折扣率的解释。视频挖掘的 ROI 是多少？把你的分析尽力限制在 15 页以内。
- 4. 为在儿童的 R U 中进行视频挖掘投资开发一个项目计划和进度可行性研究。确保包括干特图和 PERT/CPM 图，以及对要完成的所有任务进行了的明确讨论。

团队和个人练习

- 1. 你时常认为法律问题在项目成功中扮演一份角色吗？考虑一个潜在的好的信息系统的例子，这个系统由于法律需求而受到约束或者不可行。
- 2. 作为一个团队，脑力激荡一些方法提高雇员对新

信息系统或业务过程的变更接受力。

- 3. 考虑一个业务过程改进比业务过程再工程更合适的例子。在班上讨论。

需求获取的调查研究技术

本章概述和学习目标

有效的调查研究技术对于系统项目开发十分重要。在本章中，你将学习获取和分析需求所需的工具和技术，以及如何使用各种调查研究技术收集系统的有关问题、机会和指示的信息。本章将介绍以下内容：

- 定义系统需求，区别功能需求和非功能需求。
- 理解问题分析活动，建立鱼骨图以辅助问题解决。
- 理解需求管理的概念。
- 确定7种调查研究技术，描述每种技术的优缺点。
- 理解关于有效地聆听的6项指南。
- 理解什么是肢体语言和空间关系学，以及为什么系统分析员应该了解它们。
- 描述JRP会议的典型与会者，并描述他们的角色。
- 完成JRP会议的计划过程，包括选择和配备会议地点、选择与会者和准备议程表。
- 描述使用JRP作为调查研究技术的几个优点。
- 描述将需要你同最终用户一起花大量时间来执行的一种调查研究策略。

本章关键术语

需求获取（requirements discovery）包括系统分析员用来从用户团体那里确定或提取系统问题和方案需求的那些技术。

功能需求（functional requirement）是说明信息系统必须实现什么的需求。

非功能需求（nonfunctional requirement）是说明信息系统必须具备的属性或质量的需求。

系统需求（system requirement）是信息系统必须实现的或者必须具备的属性，也称为业务需求。

鱼骨图（Ishikawa diagram）是一种用于确定、探索和描述问题及其原因和结果的图形工具，它经常被称为因果图（cause-and-effect diagram）或鱼骨图（fishbone diagram）（因为它像一个鱼骨）。

调查研究（fact-finding）是使用研究、面谈、调查表、抽样以及其他技术来收集关于问题、需求和偏好信息的正式过程。它也称为信息收集或数据收集。

需求定义文档（requirements definition document）是一种正式的文档，以便与主要关联人员就提议系统的需求进行沟通，可作为系统项目的一个约定。同义词包括需求陈述、需求规格说明和功能规格说明。

需求管理（requirements management）是管理需求的变化过程。

抽样（sampling）是收集文档、表和记录中有代表性的样本的过程。

随机抽样（randomization）是一种抽样技术，其特点是选择样本数据时没有预定的模式或计划。

分层抽样（stratification）是一种系统化的抽样技术，它试图通过将抽样分散（例如，按照公式选择文档或记录）并通过避免过高或过低的估计来减少估计方差。

观察（observation）是一种调查研究技术，系统分析员或者参与到活动中，或者观察他人执行活动来了解系统。

工作抽样（work sampling）是一种调查研究技术，它包括以随机间隔进行的大量观察。

调查表（questionnaire）是一种具有特殊目的的文档，分析员可以使用它从回答者那里收集信息和观点。

自由格式调查表（free-format questionnaire）为回答者提供了很大的回答范围。它提出一个问题，然后回答者在这个问题后面提供的空白区里填写答案。

固定格式调查表（fixed-format questionnaire）由需要从预先定义的答案中做出选择的问题构成。

面谈（interview）是一种调查研究技术，系统分析员借此通过面对面的交互从个人那里收集信息。

非结构化面谈（unstructured interview）仅仅用一个头脑中的一般目标或目的以及几个（如果有的话）特定的问题进行组织。接见者期望被接见者提供一个框架并引导谈话过程。

结构化面谈（structured interview）在面谈中，接见者有一套专门的问题询问被接见者。

开放式问题（open-ended question）允许被接见者以任何认为是合适的方式回答。

封闭式问题（closed-ended question）把回答严格限制在特定的选择范围内，或者限制为简短的直接回答。

肢体语言（body language）是沟通中的非口头信息。

空间关系学（proxemics）是人与围绕其空间之间的关系的学问。

联合需求计划（Joint Requirement Planning, JRP）是一个过程，其中高度结构化的小组会议被用来分析问题并定义需求。

头脑风暴（brainstorming）是一种用于在小组会议期间产生想法的技术。参与者被鼓励在短时间内产生尽可能多的想法，而且在所有的想法都提出之前不进行任何分析。

5.1 需求获取简介

在第 2 章中，我们讨论了系统开发的几个阶段。每个阶段对于有效地设计、构造并最终实现一个满足用户（关联人员）需求的系统都是重要和必要的。但为了开发这样一个系统，我们首先必须能够正确地确定、分析和理解用户的需求，或者说，用户想让这个系统做什么。系统分析员用来确定、分析和理解系统需求的过程和技术称为**需求获取**。需求获取主要涉及系统分析员，他们同系统用户和系统所有者一起工作，在系统开发的早期阶段获取对信息系统的业务需求的详细理解。

什么是系统需求？**系统需求**定义了信息系统必须实现的功能，或者系统必须具备的属性或质量。说明信息系统必须实现什么的需求通常称为**功能需求**，说明信息系统必须具备的属性或质量的需求通常称为**非功能需求**。

第 2 章中介绍的 PIECES 框架（见表 5-1）提供了一个很好的分类非功能需求的工具。对各种类型的需求进行分类使得类似的需求可以组织起来达到汇报、跟踪和验证的目的，还可对确定可能被忽略的需求起到辅助作用。

表 5-1 系统需求的 PIECES 分类

非功能需求类型	解 释
性能	性能需求表示为了满足用户的需要而要求系统展示的性能 <ul style="list-style-type: none">• 可接受的吞吐率是多少• 可接受的响应时间是多少
信息	信息需求表示有关用户的信息，形式为内容、时间性、正确性和格式 <ul style="list-style-type: none">• 需要的输入和输出是什么？它们必须在什么时候发生• 需要的数据要被存储在哪里• 信息必须有多及时• 对外部系统的接口是什么
经济	经济需求表示系统对减少开支或增加收益的需要 <ul style="list-style-type: none">• 必须减少开支的系统领域是什么• 应该减少多少开支或者增加多少收益• 预算限度是多少• 开发的时间表是什么
控制（和安全）	控制需求表示系统必须在其中运行的环境以及必须提供的安全类型和程度 <ul style="list-style-type: none">• 对系统或信息的访问必须被控制吗• 对隐私有什么要求• 数据的重要程度要求对数据进行特殊处理（例如备份、脱机存储等）吗
效益	效益需求表示系统以最低成本产生输出的能力 <ul style="list-style-type: none">• 在过程中有必须被消除的重复步骤吗• 在系统使用其资源的方式中存在降低成本的方法吗

(续)

非功能需求类型	解 释
服务	服务需求表示使系统可靠、灵活和可扩充的需要 <ul style="list-style-type: none">• 谁将使用系统？他们都在哪里• 有不同类型的用户吗• 相应的人员因素是什么• 在系统中需要包括什么培训设备和培训材料• 有哪些培训设备和培训材料需要独立于系统进行开发和维护？例如独立的计算机培训（CBT）程序或数据库• 对可靠性/可用性有什么要求• 应该如何包装和分发系统• 需要哪些文档

基本上说，需求获取和管理的目的是为一个新系统的用户正确地确定“知识”、“过程”和“通信”方面的需求。如果没有正确地确定系统需求，可能会导致许多问题。费用的影响可能是逐步增加的，如在以下这张由著名信息技术经济学专家 Barry W. Boehm 提供的表（见表 5-2）中所描述的一样[⊖]。他研究了几个软件开发项目，以便确定需求中那些直到开发过程后期才被发现的错误所带来的费用。

按照这些调查结果，一个直到运行阶段才被发现和改正的错误需求可能造成的费用是如果在需求阶段就发现并改正的 1 000 倍！获取需求可能会是一个费时、困难和充满失败的过程，以至于组织和个人经常寻找捷径来节省时间和费用，但是这种短视行为常常会导致前面提到的问题。现在我们已经理解了我们的目标，那么让我们来看一看实现目标的过程。

5.2 需求获取过程

需求获取包括以下活动：

- 发现和分析问题。
- 获取需求。
- 归档和分析需求。
- 需求管理。

下面我们详细地介绍每一个活动。

5.2.1 发现和分析问题

如前所述，需求解决问题。系统分析员要想获得成功，就必须熟练掌握问题分析技术。没有经验的系统分析员在试图分析问题时常犯的一个错误是把症状当成问题。结果，他们可能会设计并实现一个没有解决真正问题或者可能引起新问题的方案。开发团队使用鱼骨图来确定、分析和解决问题。鱼骨图是由 Kaoru Ishikawa 发明的，他率先在日本的川崎造船厂实施了质量管理过程并成为现代管理之父。

绘制鱼骨图是把感兴趣的问题的名字输入在图的右边（或鱼头），问题可能的原因是绘制成从主脊椎分叉的骨头。一般情况下，将这些骨头标记成 4 个基本类别：材料（material）、机器（machine）、人力（manpower）和方法（method）（4 个 M），也可以使用其他名称。替代的或额外的类别包括地点（place）、程序（procedure）、策略（policy）和人（people）（4 个 P），或者环境（surrounding）、供应商（supplier）、系统（system）和技能（skill）（4 个 S）。

关键是要有 3 ~ 6 个类别覆盖所有可能的原因域。在本章后面定义的集体讨论（brainstorming）技术常常用来给主要的骨头添上原因。当鱼骨都填上原因时，它就描述了问题所有可能的根本原因。然后开发团队就可以使用这张图来讨论并研究确定问题最可能的原因以及应该如何应对。图 5-1 描述了音阶公司的“会员拒绝履行合同”问题的鱼骨图，这张图把要解决的问题放在了最右边的方框中，已经被

表 5-2 修改一个错误的相对费用

发现错误的阶段	费用率
需求	1
设计	3 ~ 6
编码	10
开发测试	15 ~ 40
验收测试	30 ~ 70
运行	40 ~ 1 000

⊖ Donald C. Gause and Gerald M. Weinberg, *Exploring Requirements: Quality before Design* (New York: Dorset House Publishing, 1989), pp. 17-18.

确定为原因类别的5个域（人-会员、方法、合同、材料和策略）列在了使用指向鱼脊的箭头（骨头）连接的鱼骨架的方框中，对于每类问题的实际原因描述成指向类别箭头（骨头）的箭头。

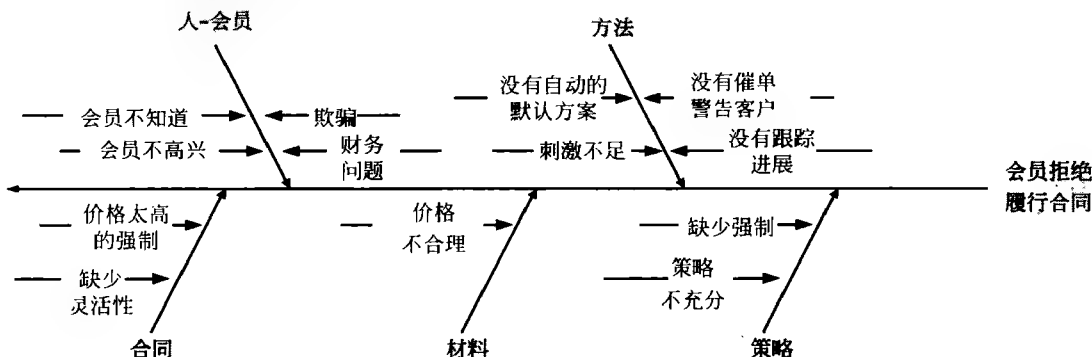


图 5-1 音阶公司案例鱼骨图

5.2.2 获取需求

给出对问题的理解，现在就可以开始定义需求。对于如今的系统分析员来说，为了成功地定义系统需求，他们必须熟练掌握收集信息的有效方法——调查研究。调查研究在整个开发周期中都要使用，但在需求分析阶段特别重要。一旦调查研究结束，工具（例如数据、过程和对象模型）将最终被用来记录事实和从事实中得出的结论。读者将在本书的后续章节学习这些工具，以及如何使用它们记录从调查研究中获得的需求。

事实属于企业应用系统及其最终用户的范畴。所以，分析员必须收集事实，以便有效地应用文档工具和技术。在系统分析阶段，分析员应了解一个企业和系统的术语、问题、机会、约束条件、需求和优先次序。

我们必须收集哪类事实呢？幸好，不管项目是什么，都有一个框架帮助我们确定需要收集什么事实。描述任何信息系统的事实也可以很好地对应到“主页”的构件上。注意调查研究技术用于系统开发的早期阶段，确定信息、功能和通信范围和目标，以及业务知识过程和系统的通信需求。

5.2.3 归档和分析需求

当执行调查研究活动时，系统分析员必须以一种有组织的、可理解的和有意义的方式归档信息（或需求草案）。这些初始文档将为系统分析员使用的建模技术提供指南，系统分析员使用这些建模技术分析需求，以便确定项目的确切需求。一旦需求确定下来，系统分析员就在文档中将那些需求形式化，用户则将检查并批准这个文档。

5.2.3.1 记录需求草案

系统分析员使用各种工具以草稿的形式记录他们的初始调查结果。编写用例来描述系统的功能，这些系统功能是从外部用户的角度来看待的，并按照外部用户可以理解的方式和词汇进行编写。决策表用来记录一个组织的复杂业务策略和决策规则，需求表则用来记录每个特定的需求。这些工具都将在本章后面更详细地介绍。

5.2.3.2 分析需求

因为需求有许多不同的来源，而且每个人对新系统的功能和特征都有自己的观点和期望，所以调查研究活动通常会产生互相矛盾的需求。需求分析的目标就是发现和解决需求中的这些问题并对修改达成一致意见，以使关联人员感到满意。这个过程涉及从关联人员那里收集的初始需求，这些需求通常是不完全的并且以一种非正式的方式记录在某些文档中，例如用例、表和报告。活动的焦点是要对关联人员的需要达成一致意见；换句话说，它应该回答这个问题：“我们获得了该项目的确切系统需求吗？”这些需求草案不可避免地会有许多错误或问题，例如：遗漏的需求、矛盾的需求、不可行的需

求、重叠的需求、二义性的需求。这些类型的需求错误在如今撰写的许多需求文档中都很常见，如果不解决这些需求错误，在后来的开发周期中再修改它们代价可能是极高的。

本章主要关注需求的业务方面（即逻辑需求），但是存在技术需求也是不争的事实。技术需求的例子包括说明一个需要的软件包或者一个硬件平台。这类需求将在第 10 章将深入讨论。

5.2.3.3 形式化需求

系统需求通常以一种正式的方式记录，以便与主要关联人员沟通。这份文档作为系统所有者和开发团队之间的约定，解释新系统要提供的内容。因此，在每个人都同意和认可文档的内容之前，该文档可能会经过许多次的改版和检查。这份文档没有标准的名称或格式，许多组织使用各种不同的名称，例如：需求陈述、需求规格说明、需求定义和功能规格说明等，而且格式通常按照组织的要求进行剪裁。那些给美国政府提供信息系统和软件的公司被要求使用政府颁布的标准文档 MIL-STD-498 中的格式和命名规则^①。许多组织已经建立了各自的标准，因为 MIL-STD-498 标准的全面性，也因为许多人都已经熟悉它了，所以各个组织的标准大多数是由它改编得到的。在本书中，我们将使用需求定义文档。图 5-2 提供了一个提纲示例。这份文档将同其他项目信息合并形成需求陈述，这是需求分析阶段最终的交付成果。

需求定义报告	
1. 引言	2.1 系统目标
1.1 目的	3. 需求和约束条件
1.2 背景	3.1 功能需求
1.3 范围	3.2 非功能需求
1.4 定义、同义词和缩略语	4. 结论
1.5 参考资料	4.1 主要问题
2. 项目概述	附录（可选）

图 5-2 需求定义提纲示例

5.2.4 需求管理

在项目的生命期间，即使需求定义文档已经被批准了，新需求的出现以及现有需求的改变仍是很常见的。某些研究已经表明，在系统投入运行以前，有 50% 或者更多的需求将发生变化。显然，这对开发团队来说将是一个十分头痛的问题。为了帮助缓解可能引起的众多问题，进行需求管理是必要的。需求管理涉及需求变化如何被处理的策略、规程和过程。它规定了应该如何提交一个需求变化请求；如何分析需求变化对范围、进度和费用的影响；如何批准或驳回需求变化；如果批准了需求变化，应该如何实现。

5.3 调查研究技术

本节介绍 7 种常用的调查研究技术：

- 对现有文档、表和数据库进行抽样。
- 调研和实地访问。
- 观察工作环境。
- 调查表。
- 面谈。
- 原型化。
- 联合需求计划。

分析员在一个系统项目开发期间通常会应用其中的多项技术。为了能够在任何情况下选出最合适的技术，你需要了解每项技术的优点和缺点。

5.3.1 对现有文档、表和文件进行抽样

当研究一个现有系统时，系统分析员可以通过研究现有文档、表和文件建立对该系统的感性认识。一个优秀的分析员总是首先从现有文档中获取事实，而不是首先从别人那里获取事实。

① MIL-STD-498 是一个标准，它合并了两个标准：DOD-STD-2167A 和 DOD-STD-7935A，定义了一套既适用于武器系统开发也适用于自动化信息系统开发的活动和文档。

5.3.1.1 从现有文档收集事实

分析员应该找出的第一份文档是组织结构图。组织结构图提供了该项目的关键所有者和用户，以及他们之间的组织关系。分析员还可能想跟踪产生这个项目的历史。为了实现这个目标，分析员需要收集并检查描述问题的文档，例如：

- 办公室之间的便函、研究、琐事、建议箱、客户抱怨和记录问题的报告。
- 财务记录、工作成绩回顾、工作度量回顾以及其他安排的工作报告。
- 信息系统项目请求——过去的和现在的。

除了描述问题的文档以外，通常还包括描述正被研究或设计的业务功能的文档，例如：

- 公司任务陈述和战略计划。
- 正被研究的下级部门的正式工作目标。
- 可能对建议的系统产生约束的政策条款。
- 用于特定日常操作的标准操作规程（Standard Operating Procedure, SOP）、工作概要或任务指令。
- 在过程周期的各个点上代表实际事务的完整表格。
- 手册和计算机数据库的样本。
- 手册和计算机屏幕及报表的样本。

另外，分析员还经常检查由系统分析员和顾问以前做的系统研究和设计文档，这些文档可能包括：

- 各种类型的流程图和图形。
- 项目字典或资料库。
- 设计文档，例如输入、输出和数据库。
- 程序文档。
- 计算机操作手册和培训手册。

所有收集到的文档都应该被分析，以确定信息的时效性。但不要丢掉过期的文档，时刻记着需要额外的调查研究来验证或修改收集到的事实。分析员在所有这些材料中寻找什么？从这些文章中收集到的东西包括：

- 问题的症状和可能的原因。
- 企业中什么人问题有所理解。
- 支持目前系统的业务功能。
- 需要由系统收集和报告的数据的类型。
- 文档中分析员不理解的东西，以及需要在面谈中涉及的东西。

5.3.1.2 文档和文件抽样技术

因为研究一个文件或数据库中出现的每个表或每个记录是不切实际的，所以系统分析员通常使用抽样技术获得一个足够大的样本集，以确定系统中可能会发生什么。系统分析员力图抽样足以表示数据的全部特性和复杂性的表。有经验的分析员避免抽样空表——空表说明不了这个表如何使用、不能如何使用以及错误使用的信息。当研究文档或者来自数据库表的记录时，应该研究足够多的样本，确定所有可能的过程条件和例外条件。可以使用统计抽样技术来确定样本大小是否足以具有代表性。

由于存在众多的抽样问题和因素，所以读者应该选修一门介绍性的统计课程。用于确定样本大小的一个简单而有效的公式是：

$$\text{样本大小} = 0.25 \times (\text{确定性因子} / \text{可接受的错误})^2$$

确定性因子取决于你希望抽样数据包括了样本中的各种情况有多大的确定性。确定性因子是从表中计算出来的（这个表可以在许多工业工程课本中找到），表 5-3 是这张表的一部分。

假设你希望发票样本包含的所有情况具有 90% 的确定性，那么你的样本大小（SS）计算如下：

表 5-3 确定性因子表的一部分

期望的确定性	确定性因子
95%	1.960
90%	1.645
80%	1.281

$$SS = 0.25 \times (1.645/0.10)^2 = 68$$

为了得到期望的确定性，我们需要抽样 68 张发票。如果想得到更高的确定性，我们必须抽样更多的发票。

现在假设我们从经验中得知每 10 张发票中就有 1 张发票与常规情况不同。根据这条知识，我们可以修改上面的公式，将启发式因子 0.25 替换成 $p(1-p)$ 。其中， p 是有差异发票的比率。

$$SS = p(1-p)(1.645/0.10)^2$$

通过使用这个公式，我们可以减少为了得到期望的确定性所需的样本数量。即：

$$SS = 0.10 \times (1 - 0.10) \times (1.645/0.10)^2 = 25$$

如何选择这 25 张发票呢？两种常用的抽样技术是随机抽样和分层抽样。随机抽样随机选择采样数据。因此，我们只是根据上面计算出的样本大小随机地选取 25 张发票。分层抽样是一种有考虑的系统的方法，试图降低采样数据的方差。对于计算机文件来说，分层抽样可以通过编写一个简单的程序来实现。例如，假设我们的发票存储在一个拥有大约 250 000 张发票的数据库中，由于样本大小需要包括 25 张发票，所以可以编写一个程序每隔 10 000 条记录（= 250 000/25）打印一条记录。对于手册文件和文档，我们将执行类似的抽样策略。

5.3.2 调研和实地访问

第二种调查研究技术是全面地研究问题领域。大多数问题都不是独一无二的，其他人在我们之前可能已经解决了这些问题。组织经常联系或实地考察对类似问题有经验的公司，如果这些公司愿意共享信息，他们就有可能获得有价值的信息，这可以节省开发过程中的大量时间和费用。

计算机期刊杂志和参考书也是一个很好的信息来源。它们可以为你提供有关他人如何解决类似问题的信息。现在由于互联网的发展，你甚至足不出户就可以进行调研。通过你的个人计算机探索因特网你能获得不可限量的信息。

类似的调研例如访问已经解决过类似问题的其他公司或部门。专业社团（例如信息技术专家联合会 [AITP] 或者信息系统联合会 [AIS]）可以为成员提供一个有用的联系网络。

5.3.3 观察工作环境

为了获得对系统的理解，观察是一种有效的数据收集技术。观察要求系统分析员成为人员和活动的观察者，以便于了解系统。当通过其他方法收集的数据的有效性值得怀疑时，或者当系统某方面的复杂度妨碍了用户做出清晰的解释时，就经常使用这项技术。

5.3.3.1 通过观察工作中的人收集事实

如果你有能力全面而且正确地观察的话，观察可能是一种十分有用而且有益的调查研究技术。观察的优缺点如下：

优点

- 通过观察收集的数据可能十分可靠。有时观察被用来检查直接从个人那里获得的数据的有效性。
- 系统分析员能够确切地明白要做什么。复杂的任务有时难以清楚地用语言解释，因此只能求助于观察。通过观察，系统分析员可以确定那些被忽略了的任务或者被其他调查研究技术错误地描述的任务。而且，分析员可以获得描述任务的物理环境数据（例如，物理布局、交通、照明、噪声等级）。
- 同其他调查研究技术相比，观察技术相对廉价。其他技术通常要求更多的雇员脱产，从而花费更多的时间和费用。
- 观察技术使得系统分析员可以进行工作度量。

缺点

- 因为人们在被观察时通常会觉得不舒服，所以当被观察时可能会不自觉地表现得与平常不一样。
- 正被观察的工作可能不具有通常在那个时间段期间应该经历的工作难度或工作量等级。
- 某些系统活动可能在特别的时间发生，不便于系统分析员进行观察。

- 正被观察的任务容易受到各种类型的打断。
- 某些任务可能不总是以它们被系统分析员观察到的方式执行。例如，系统分析员可能已经观察了一个公司如何履行几张客户订单。但是，观察到的工作规程可能是用来履行一些常规客户订单的步骤。如果那些订单中有一些特殊订单（例如，一份购买一般不保存在库房中的商品的订单），系统分析员将观察到一套不同的工作规程。
- 如果人们已经在以违反标准操作规程的方式执行任务，当你观察他们时，他们可能会临时性地以正确的方式干活。换句话说，人们可以让你看到他们想让你看到的东西。

5.3.3.2 对观察的指南

系统分析员如何通过观察获得事实？是一个人简单地来到观察地点然后开始记录他看到的每件事吗？当然不是。首先要做大量的准备工作。分析员必须决定如何实际地收集数据。需要快速记录数据的特殊表格吗？正被观察的人会因为有人观察并记录其行动而感到厌烦吗？什么时间是被观察的任务的空闲、一般和高峰时间？系统分析员必须确定观察系统的某个特定方面的理想时间。

观察应该首先在“典型”负载情况下进行。之后，观察可以在峰值期间进行，以收集用于度量由于工作量的增加而产生的影响的信息。系统分析员也可以对那些被观察的系统将使用的文档或表格进行抽样。

对于观察来说，前面讨论的抽样技术也很有用。在这种情况下，抽样技术称为工作抽样，其中大量观察的以随机间隔进行。这种技术对正被观察的人的威胁较小，因为观察时间是不连续的。当使用工作抽样时，需要预先定义被观察的工作的操作内容。然后计算一个抽样大小，就像你在文档和文件抽样中做的那样。最后，进行多次随机的观察，注意观察每天不同时间的活动。通过记录观察期间每个操作出现的次数，你将建立雇员每天如何使用时间的感性认识。

5.3.3.3 现场直播系统

在这种类型的观察中，系统分析员主动地扮演一小段时间的用户角色，这是了解系统问题和需求最有效的方法之一。通过站在用户的角度看问题，系统分析员可以快速地获得对用户经验及其工作的理解。这种类型的角色扮演向系统分析员提供了有关业务过程和功能以及同它们相关的问题和挑战的第一手资料。

5.3.4 调查表

另一种调查研究技术是通过调查表进行调查。可以大量印刷该文档并分发给回答者，然后回答者可以在自己有时间的时候填写调查表。调查表使得系统分析员可以从一大群人那里收集事实，同时保持统一的答复。当面对大量观众时，其他调查研究技术都不能如此有效地将同样规模的事实表格化。

5.3.4.1 通过使用调查表收集事实

系统分析员经常批评调查表的使用。许多系统分析员声称回答缺少可靠性和有用信息。但是调查表仍是一种有效的事实收集方法，对调查表的这些批评许多都可以归因于系统分析员不合适或低效地使用调查表。在使用调查表之前，应该首先理解使用调查表的优点和缺点。

优点

- 大多数调查表可以得到快速的回答。人们可以在他们方便的时候完成和返回调查表。
- 调查表提供了一种可以从大量的人群中收集数据的相对廉价的方法。
- 调查表允许回答者匿名填写。所以，人们会更愿意提供真实数据，而不是告诉你他们认为老板想让他们说的数据。
- 回答可以快速地表格化和分析。

缺点

- 回答者的数量经常很低。
- 无法保证个人会回答或者进一步说明所有问题。
- 调查表往往不灵活。没有给系统分析员提供机会获得来自回答者自愿提供的信息，或者重新说明可能已经被错误解释的问题。

- 系统分析员不可能观察和分析回答者的肢体语言。
- 没有机会立即澄清对问题的含糊或不完整的回答。
- 好的调查表很难准备。

5.3.4.2 调查表的类型

调查表有两种格式：自由格式和固定格式。自由格式调查表设计成让用户可以更自由地回答每个问题。

下面是使用自由格式调查表的两个例子：

- 你目前收到了什么报告？它们是如何使用的？
- 这些报告有问题吗？例如，它们是否正确？是否信息不足？或者它们是否难以阅读或使用？如果是这样，请解释。

第二种类型的调查表是固定格式调查表。固定格式调查表比较严格，要求回答者从预先定义的答案中做出选择。对于任何问题，回答者必须从可选答案中进行选择。这使得结果更容易表格化，但是回答者可能无法提供有价值的额外信息。

固定格式的问题有下列3类。

1. 多项选择问题给回答者提供几个答案。回答者应该被告知是否可以选择多个答案。当没有一个标准答案可用时，有些多项选择问题允许提供很简明的自由格式回答。固定格式多项选择问题的例子如下：

你觉得退单出现得太频繁了吗？

☐是 ☐不是

你收到的当前应收账款报告有用吗？

☐有用 ☐没用

如果没用，请解释原因。

2. 分级评定问题给回答者提供一段话，然后要求回答者使用提供的答案陈述一个观点。为了防止问题自身的偏好，问题中应该有同样数量的正面和反面答案。下面是固定格式分级评定问题的例子：

实现数量折扣将会增加客户订单。

☐强烈同意

☐同意

☐不表态

☐不同意

☐强烈反对

3. 次序评定问题给回答者提供几个可能的答案，每一个都要求按照喜好或经验进行排序。固定格式次序评定问题的例子如下：

按照你花的处理时间，对以下事务进行排序：

_____ % 新客户订单

_____ % 订单取消

_____ % 订单修改

_____ % 支付

5.3.4.3 制作调查表

好的调查表是“设计”出来的。如果你不首先设计就编写调查表，就会降低成功的机会。以下是有效的调查表制作规程：

1. 确定必须收集什么事实和观点以及你应该从谁那里收集。如果对象人群的数量很大，考虑使用一个较小的人群，随机选择一组回答者。
2. 根据需要的事实和观点，确定是自由格式问题还是固定格式问题能产生最佳答案。经常使用的是一种具有可选自由格式解释的固定格式问题。

3. 编写问题。检查它们的结构错误及可能的错误解释，确保问题中没有反映你个人的偏好和观点，然后编辑问题。

4. 在一个小的回答者样本中测试这些问题。如果你的回答者觉得有错误，或者如果答案没用，则重新编辑这些问题。

5. 复制并分发调查表。

5.3.5 面谈

一般地，个人面谈是最重要和最常用的调查研究技术。个人面谈通过直接、面对面的交互获取需求。面谈可以用来实现下列目标：发现事实、验证事实、澄清事实、激发热情、让最终用户参与、确定需求以及征求想法和观点。在一次面谈中有两个假定的角色。系统分析员是接见者（interviewer），负责组织和引导面谈；系统用户或系统所有者是被接见者（interviewee），要求他们回答一系列问题。

可以同时有一个或多个接见者和/或被接见者。换句话说，面谈可以一对一地进行，也可以多对多地进行。遗憾的是，许多系统分析员是差劲的接见者。本小节将讲述如何正确地进行面谈。

5.3.5.1 通过与用户面谈收集事实

信息系统中最重要要素是人。没有哪种调查研究技术像面谈一样对人如此重视，但不同的人有不同的价值、优先权、观点、感情和个性。所以，为了使用面谈技术，你必须拥有良好的人际关系能力，以便有效地应对不同类型的人。像其他调查研究技术一样，面谈也不是在所有情况下都最合适。面谈也有它的优点和缺点，这些优缺点应该与其他调查研究技术的优缺点进行权衡。

优点

- 面谈为分析员提供了激发被接见者自由开放地回答问题的机会。通过建立相互之间的友善关系，系统分析员能够给被接见者一种主动为系统项目做出贡献的感觉。
- 面谈使系统分析员从被接见者那里得到更多的反馈。
- 面谈使系统分析员对每个人调整或重述问题。
- 面谈为系统分析员提供观察被接见者的以非语言形式表达问题的机会。一个优秀的系统分析员可以通过观察被接见者的肢体动作和面部表情以及通过聆听对问题的口头回答获得信息。

缺点

- 面谈非常耗时，因此费用高昂。
- 面谈的成功极大地取决于系统分析员的人际关系能力。
- 面谈可能会由于被接见者的地理位置而变得不现实。

5.3.5.2 面谈的类型和技术

有两种类型的面谈：非结构化面谈和结构化面谈。非结构化面谈的特点是涉及一般性的问题，被接见者可以引导谈话过程。这种类型的面谈常常会偏离主题，分析员必须准备好重新将面谈引导回主要目标或主题。因此，非结构化面谈对于系统分析和设计来说通常不太合适。结构化面谈要求接见者询问一套专门设计用于从被接见者处获取特定信息的问题。根据被接见者的回答，接见者将提出额外的问题以获得澄清或进一步深入。这些问题中有一些可能是计划中的，其他一些则是自然产生的。

非结构化面谈常常询问开放式问题。这类问题赋予被接见者很大的回答空间。一个开放式问题的例子是：“你为什么对不可汇集账号报告不满意？”结构化面谈常常询问更加封闭式问题，这些问题设计成从被接见者处获得简短的直接的回答。这样问题的一个例子是：“你及时收到了不可汇集账号报告吗？”或者“不可汇集账号报告包含了正确的信息吗？”现实情况下，大部分问题位于两种极端情况之间。

5.3.6 如何进行面谈

作为系统分析员，你的成功至少部分地取决于你的面谈能力。一次成功的面谈涉及选择合适的面谈对象、大量的面谈准备、正确地进行面谈以及面谈的后续工作。下面我们就更详细地介绍这些步骤。

假定你已经确定了需要一次面谈，并且已经明确了需要获得哪类事实和观点。

5.3.6.1 选择被接见者

你应该同你正在研究的信息系统的最终用户面谈。一个正式的组织结构图将有助于确定面谈对象及其职责。你应该在面谈之前尽可能多地了解每一个人，试着了解他们的实力、害怕的事情、偏见和动机，然后就可以在面谈中考虑这些人的特点。

总是事先预约被接见者，绝不要直接闯进，将约会地点限制在半个小时至一个小时可到达的地方。被接见者的管理级别越高，你安排的面谈时间就应该越短。如果被接见者是一个办事员、服务员或蓝领工人，在安排面谈之前要首先获得其主管的同意。确定你希望进行面谈的地点在面谈安排的时间内可用。不要在你的办公室同事或者与被接见者同级的人在场的情况下进行面谈。

5.3.6.2 准备面谈

准备是面谈成功的关键。一个被接见者很容易发现接见者没有准备好，并且可能对缺乏准备产生不满，因为这浪费了他的宝贵时间。当进行约会时，应该向被接见者通报谈话主题。为了确保主题的所有有关方面都被涉及到，分析员应该准备一份面谈指南。面谈指南是接见者将询问被接见者的问题清单。面谈指南也可以包含进一步的问题，这些问题只有在对其他问题的回答需要额外解释时才被问及。图 5-3 显示了一份面谈指南的例子。表中的议程被仔细地安排，其中每个问题都分配了相应的时间，也应该预留出提出进一步的问题和重新回到面谈的时间。问题应该仔细地选择和遣词造句，大多数问题都以标准的谁、什么、什么时候、在哪里、为什么以及如何之类的词开始。注意，不要使用下列类型的问题：

- 含沙射影的问题，例如，“我们不得不需要报告中所有这些列吗？”这个提问传达了接见者对这个问题的个人观点。
- 对答案有诱导性的提问，例如，“你不准备使用这个 OPERATOR CODE，是吗？”这个提问引导被接见者回答：“不，当然不”，而不论其实际观点是什么。
- 有偏见的问题，例如，“我们需要多少位编码用于 INVENTORY FILE 中的 FOOD CLASSIFICATION？我想 20 应该足够了。”为什么要使被接见者的回答带上你自己的偏见呢？

接见者应该特别避免胁迫式或批评式的提问。面谈的目的是研究，而不是评价和批评。面谈提问指南如下：

- 使用清楚精确的语言。
- 不要把你的观点作为提问的一部分包括进去。
- 避免冗长而复杂的问题。
- 避免胁迫式的问题。
- 当你的意思是指一群人时，不要使用“你”。

5.3.6.3 进行面谈

尊敬你的面谈对象及其时间。穿着要与面谈对象相配。这意味着你同经理面谈时一般来说穿着将与码头工人面谈时穿着不同。如果面谈在会议室举行而不是面谈者的办公室，早点到以确保会议室准备好了。

通过事先感谢面谈对象开始面谈。说明面谈的目的和长度，以及收集的数据将被如何使用。然后监控时间以便你能信守诺言。

询问后续的问题。不断试探直到你理解了系统需求。特别要询问例外条件。对于“如果——什么”类型的问题，例如：“如果不清楚怎么办？”或者“如果产品不在库存中会发生什么？”仔细地倾听并观察被接见者，留心来自被接见者的以语言方式和非语言方式给出的回答。对你来说保持面谈按计划进行很重要，还要预测是否需要调整与被接见者的面谈内容。如果某个问题已经在前面的另一个问题中回答了，这个问题经常可以省略掉；或者如果根据你在面谈期间已经了解到的情况，某个问题被认为是无关的问题，可以删除该问题。

被接见者: Jeff Bentley, 应收账款经理 日期: 2003 年 1 月 19 日 时间: 下午 1:30 地点: Admin. Bldg. 223 房 主题: 目前的信用检查策略		
分配的时间	接见者的问题或目标	被接见者的回答
1~2 分钟	目标 开始面谈: • 自我介绍 • 感谢 Bentley 先生的到来 • 陈述面谈目的——为了了解现有信用检查策略	
5 分钟	问题 1 什么条件决定了一个客户的订单是否通过信用认可 进一步询问的问题	
5 分钟	问题 2 一旦已经评估了这些条件, 最可能采取的决策或行动是什么 进一步询问的问题	
3 分钟	问题 3 当客户订单的信用没有得到认可时, 如何通知客户 进一步询问的问题	
1 分钟	问题 4 当一个新订单获得了信用认可并放在了包含可以履行的订单文件中后, 客户可能会请求修改订单。如果新的订单总额超过了原先的总额, 订单要再次进行信用认可吗 进一步询问的问题	
1 分钟	问题 5 进行信用检查的人是谁 进一步询问的问题	
1~3 分钟	问题 6 可以允许我同那些人谈话以专门了解如何实施信用检查过程吗 进一步询问的问题 如果允许: 什么时候同他们会面合适	
1 分钟	目标 总结面谈: • 感谢 Bentley 先生的合作并告诉他将送给他一份面谈内容的副本	
21 分钟	为基本问题和目标分配的时间	
9 分钟	为进一步询问的问题和重新回到主题分配的时间	
30 分钟	为面谈分配的总时间 (下午 1:30 ~ 2:00)	
一般评价和注释:		

图 5-3 面谈指南范例

下面是一组面谈过程中应该遵守的规则:

应该做到:

- 穿着恰当。
- 有礼貌。
- 仔细聆听。
- 保持控制。
- 探查。
- 观察特殊习惯和非口头交流。

避免:

- 假定答案存在或者不存在。
- 提示口头的或者非口头的线索。
- 使用行话。
- 显示个人偏见。
- 谈论而不是聆听。
- 对有关主题和被接见者的情况做出假定。

- 有耐心。
- 让被接见者保持放松。
- 保持自我控制。
- 按时结束
- 磁带录音——聆听能力差的表现。

在面谈结论阶段应该表达感谢并回答被接见者提出的问题。结论阶段对于保持与被接见者的亲善和信任关系是很重要的。

5.3.6.4 面谈的后续工作

为了有助于维持同被接见者良好的友善关系和信任，应该送给他们一份总结了面谈内容的备忘录。这份备忘录应该提到被接见者对项目的贡献，并给他们机会澄清可能在面谈期间得出的任何错误解释。另外，应为被接见者提供额外信息的机会，也许这些信息在面谈期间没能提供出来。

5.3.6.5 聆听

当大多数人谈论沟通技能时，他们想到的是说话和写作。聆听的能力很少被提到，但聆听可能是面谈过程中最重要的能力。为了进行一次成功的面谈，你必须区分清楚听到和聆听，“听到是意识到有人在说话，聆听是理解说话者想交流的内容。”^①

我们已经习惯于在我们生活中的大部分时间不去聆听。我们不去理会吵闹的兄弟姐妹，而独自享受喜爱的音乐 CD，或者我们通过远离分心的事情（例如聒噪的同屋）去学习。我们已经学会了不去聆听，但我们还应该学会如何有效地聆听。

当和用户一起工作解决问题时，与用户沟通可能是很困难的。下面的指南可以打开沟通的渠道：

- 带着积极的态度开会。
- 让别人放松。
- 让他们知道你在聆听。
- 问问题。
- 不要做任何假设。

5.3.6.6 肢体语言和空间关系学

什么是肢体语言？为什么系统分析员在面谈过程中关心肢体语言？**肢体语言**是人们沟通中的所有非口头信息，是我们在沟通中都使用但通常又没有意识到的一种非口头信息。

研究发现一个令人吃惊的事实：在一个人的全部感觉中，只有 7% 是通过口头交流（用语言）的，38% 是通过语调交流的，55% 是通过面部和肢体表情交流的。如果你只是听了某人的话语，你就错过了他要说的绝大部分内容！

本书中，我们将只集中介绍 3 个方面的肢体语言：面部表情流露、目光接触和姿态。面部表情流露意味着你有时可以通过观察人们脸上的表情理解其感觉。许多常见的感情都与容易辨认的面部表情相关联。但是，脸是人体中最可控的部分之一，那些知道表情经常流露出思想的人很善于控制表情。

另一种形式的非口头交流是目光接触，目光接触是脸部受控制最少的部分。你曾经有过向一个不看着你的人说话吗？那使你感觉如何？持续地缺少目光接触可能说明了不确定性。通常的一瞥常常是 3~5 秒钟，但直接的目光接触应该随着距离而增加。作为一名分析员，你需要小心但不要过度地使用带有威胁性的目光与用户接触，以便你不会进一步地胁迫他们。直接的目光接触能够在他人身上产生强烈的感情（正面的或者负面的）。

姿态是身体中受控制最少的部分。因此，对于机敏的分析员来说，身体姿态包含了丰富的信息。具有一致意见的小组成员倾向于表现出同样的姿态。优秀的分析员会观察观众姿态的变化，姿态可能表现出忧虑、不同意或厌倦。分析员通常应该保持一种“开放”的身体姿势，发出易接近、认同和有接受力的信号。在特殊环境下，分析员可以选择使用一个面对面的正面角度或者 90 度角以对另一个人

① Thomas R. Gildersleeve, *Successful Data Processing Systems Analysis* (Englewood Cliffs, NJ: Prentice Hall, 1978), p. 93.

建立控制感。

除了通过肢体语言进行信息沟通外，人们还可通过空间关系学沟通。空间关系学（人与围绕其空间之间的关系）是沟通中的一个要素，它可以被了解它的分析员控制。

人们往往对其空间非常具有领土防御性。观察你的同班同学在一门没有指定座位的课上如何坐。或者下次你同某人谈话时，有意地移近一点或远一点，看发生什么情况。一个好的分析员能够了解空间区域。

5.3.7 获取原型

另一类调查研究技术是原型化。原型化在第3章中已介绍，用于快速应用开发（RAD）。原型化背后的概念是建立用户需求的一个小型的工作模型或者一个信息系统的建议设计。这类原型化技术通常是一种设计技术，但这种方法能够应用于系统开发生命周期的早期，以进行调查研究和需求分析。以确定需求为目的的构成原型的过程称为获取原型。

获取原型技术常常应用于系统开发项目中，特别是当开发团队难以定义需求时，其原理是当用户看到需求时他们就将意识到他们的需求。原型应该被快速地开发出来，以便可以在开发过程中使用。通常，只对那些没有被清楚地理解的需求建立原型，这意味着大量期望的功能可能没有被包括在原型中，而且可以忽略质量保证。非功能需求（例如性能和可靠性）可能没有像对最终产品那样的严格要求。通常，将使用不同于最终软件所使用的技术来构造获取原型。在这些情况下，当系统完成时，原型可能被丢弃。这种“扔掉”方式主要用于收集信息和系统概念。一个被建议的系统的许多领域可能还没有被清楚地理解，或者某些特征可能对开发人员来说是一个技术挑战，创建获取原型使得开发人员以及用户可以更好地理解 and 提炼系统中涉及的问题。这项技术尽力降低了发布一个没有满足用户需求或者不能实现技术需求的系统的风险。

获取原型技术有它的优缺点，对于每种调查研究情况，应该与其他调查研究技术的优缺点进行权衡。

优点

- 允许用户和开发人员用软件做实验并获得对系统可能如何工作的理解。
- 在高开发费用支出之前，辅助确定系统的可行性和有用性。
- 作为给用户的一种培训机制。
- 辅助构造最后在系统测试阶段使用的系统测试计划和场景。
- 可以使用于调查研究的时间最小化并有助于定义更稳定、更可靠的需求。

缺点

- 开发人员可能需要接受原型技术方法方面的培训。
- 用户可能会基于原型的性能、可靠性和特征产生不现实的预期。原型只能模拟系统功能，在本质上是完整的。必须清楚地告诉用户这个事实并不要误导他们。
- 制作原型可能延长了开发进度并增加了开发费用。

5.3.8 联合需求计划

许多组织正使用小组工作会议代替大量的独立的面谈。小组工作会议方法的一个例子是联合需求计划（JRP），其中高度结构化的小组会议被用来分析问题并定义需求。为了能按预期的那样工作，这项技术及类似技术通常需要大量的培训。但是，它们可以极大地减少在一个或多个生命周期阶段中调查研究所花的时间。JRP（和JAD）技术在系统计划和系统分析阶段越来越常用，可以用来获得小组关于问题、目标和需求的一致意见。在本小节中，你将了解到JRP会议的与会者及其角色。也将学习如何计划和主持一个JRP会议、在一个JRP会议期间使用的工具和技术以及通过JRP实现的好处。

5.3.8.1 JRP与会者

联合需求计划会议包括一些不同的参与者和角色，期望每个参与者都能够参加并主动地参与整个JRP会议。下面介绍一个典型的JRP会议中包括的人员及其角色。

- 负责人——任何成功的 JRP 会议都需要一个 JRP 负责人作为它的拥护者。这个人通常是位于顶层管理层（非 IT 或 IS 管理层）的人，并且他的职权跨越系统项目中涉及的不同部门和用户。负责人通过鼓励用户主动地参与 JRP 会议对系统项目给予完全的支持。根据系统开发的“逐步投入”方式，通常由负责人做出项目继续还是不继续进行的最后决策。
- 主持人——JRP 会议还应该包括一个扮演领导者或主持人角色的人。JRP 主持人通常负责领导一个系统项目的所有会议，这个人具有出色的沟通能力，拥有协商和解决小组矛盾的能力，拥有丰富的业务知识，具有出色的组织能力，对将做出的决策保持公平，并且不用向任何 JRP 会议与会者汇报工作。

JRP 主持人将计划 JRP 会议，主持这个会议，直到会议结束。在会议期间，主持人负责领导讨论，鼓励出席者主动参与，解决可能产生的矛盾，确保实现会议的目标和目的。建立会议期间将遵守的基本规则并确保与会者遵守这些规则是 JRP 主持人的责任。

- 用户和管理人员——联合需求计划包括一些来自用户和管理层的参与者，他们从各自每天的工作中抽出时间来参加 JRP 会议。这些参与者通常由项目负责人选择，他必须仔细地确保每个人具有在调查研究会议期间做出贡献的业务知识。项目负责人必须行使权力和鼓励，以确保这些人承诺主动参与。

一个典型的 JRP 会议可以包括一些用户和管理人员，人数为十几人或者更多。JRP 会议期间用户的角色是用来有效地沟通业务规则和需求、评审设计原型并做出是否接受的决策。JRP 会议期间管理人员是用来批准项目目标、设置项目优先权、批准进度和费用以及批准确定的培训需求和实现计划。

- 记录员——一次 JRP 会议还包括一个或多个记录员，他们负责记录会议上讨论的每一件事情。这些记录在会后立即发表并散发给与会者，以便维持 JRP 会议及其成员的动力。快速地发表记录被反映在更多的记录员使用 CASE 工具来收集 JRP 会议期间沟通的众多事实上（使用数据模型和过程模型记录）。因此，对于记录员来说，具有深入的系统分析和设计知识并熟练使用 CASE 工具是有好处的。系统分析员经常扮演这个角色。
- IT 职员——一次 JRP 会议还可能包括一些 IT 人员，他们主要聆听和记录用户和管理人员说的有关问题和需求。通常，除非被邀请发言，否则 IT 人员一般不发言。相反，他们的任何问题和关注都在 JRP 会议之后或之前不久直接提交给 JRP 主持人。JRP 主持人发起和推动用户和管理人员的问题讨论。

在 JRP 会议中，IT 职员通常由项目团队的成员组成。这些成员可以和记录员密切合作，以开发模型和有关会议期间沟通的事实的其他文档。专家也可能被召来，提供可能产生的有关特殊技术问题和担心的信息。当条件允许时，JRP 主持人可以提示 IT 专家解决技术问题。

5.3.8.2 如何计划 JRP 会议

大多数 JRP 会议需要 3~5 天时间，但有时可持续超过两周。任何 JRP 会议的成功都取决于正确的计划和有效地实施计划。在 JRP 会议开始之前，某些准备是需要做好的。在计划 JRP 会议之前，分析员必须同主要负责人密切配合，以确定要通过 JRP 会议讨论的项目范围。确定每个 JRP 会议的高层需求和预期也很重要，这通常包括与所选的人面谈，这些人负责项目涉及的部门。最后，在计划 JRP 会议之前，分析员必须确保主要负责人愿意把人员、时间和其他资源投入到这个会议中。

计划一个 JRP 会议包括 3 个步骤：选择 JRP 会议地点、选择 JRP 会议参与者、准备 JRP 会议议程。下面就详细介绍这些步骤。

选择 JRP 会议地点

当可能时，JRP 会议应该在公司工作地点以外召开。大多数本地旅馆或大学都有用来承办小组会议的设施。通过在外面举办 JRP 会议，与会者的精力集中在与 JRP 会议有关的问题和活动中，避免了在他们通常的工作地点可能出现的打断和分心。无论 JRP 会议的地点在哪里，所有的与会者都应该被要求到场并禁止回到他们通常的工作地点去。

会议室或主会议室应该舒适地坐下所有的与会者。房间应全部配备桌子、椅子和其他满足所有与会者需要的设施。图 5-4 描绘了一个典型的 JRP 会议的房间布局。JRP 会议室里典型的可视化辅助工具应该包括一个白板或者黑板，一个或几个挂纸板，一个或几个投影仪。

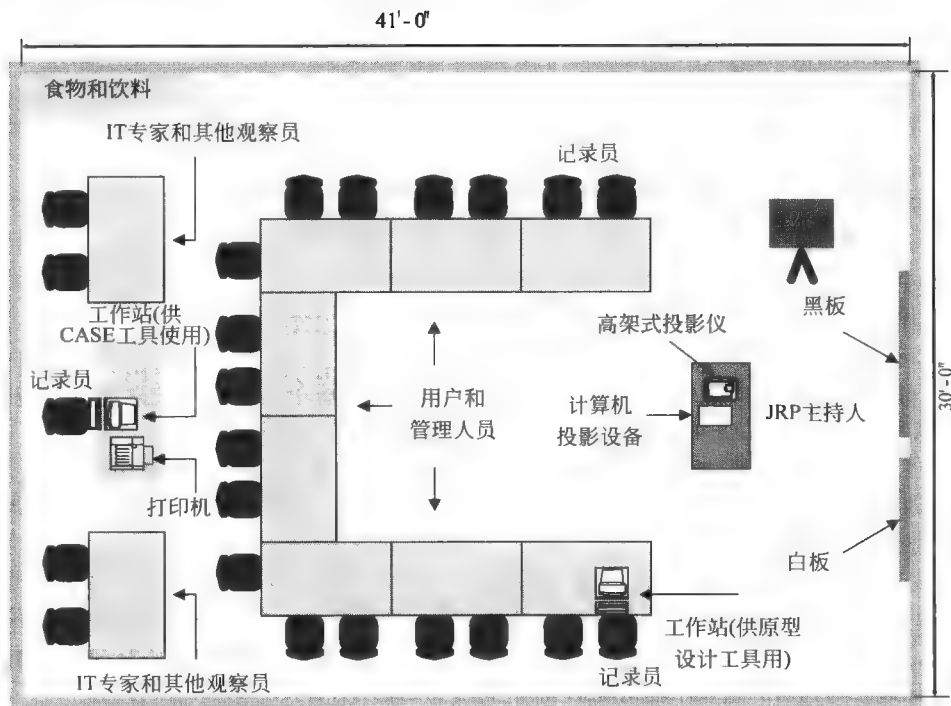


图 5-4 JRP 会议的典型房间布局

房间里还应该配备记录员记录会议期间沟通的事实和问题所需的计算机设备。计算机应该包含支持各种类型的记录或文档的软件包，这些记录或文档由记录员收集并在以后发表。这类软件可能包括 CASE 工具、字处理软件、电子表格软件、演示报告软件、原型化软件、打印机、复印机和计算机投影设施。计算机设备（除了那些用于原型化的）应该放在房间的后面，以便不会影响到与会者。与会者之间的个人交流应集中在会议内容上，而不是技术上。

最后，房间里应该配备电话会议设备以便远地用户能够参与会议。房间里应该为用户、管理人员和其他与会者提供笔记本和铅笔。还应该为与会者提供名签和座次牌。计划中还应该提供饮料和小吃，以使与会者尽可能地感到舒适。身体的舒适是很重要的，因为 JRP 会议强度很高，并经常一开就是一整天。

选择 JRP 会议参与者

如前所述，要选择与会者，这包括：JRP 主持人、记录员和用户团体的代表。用户应该是对他们的业务十分了解的关键人士。不过，经理们常常十分依靠这些人经营业务，也就常常会对把他们从工作岗位上派出来参加会议感到犹豫不决。因此，分析员必须确保管理层同意这个 JRP 项目并愿意让这些关键人物参加会议。还需要选择各种的 IT 专家参与 JRP 会议。通常分配到该项目团队的所有 IT 职员都参与 JRP 会议，还可以指定其他 IT 专家解决与该项目有关的专门技术问题。

准备 JRP 会议议程

准备是 JRP 会议成功的关键。JRP 会议主持人必须准备材料，以简要地向与会者介绍会议的范围和目标。另外，每个 JRP 会议的议程应该在每次会议之前就准备好并分发出去。议程指示了会议期间将被讨论的问题以及给每个议题分配的时间。

议程应该包含 3 个部分：开始、主体和结论。开始部分用于交流会议的预期，沟通基本规则，影响或激发与会者参与；主体部分用于细化要在 JRP 会议中涉及的主题或问题；最后，结论部分用于留出

时间总结当天的会议，提醒与会者会议未解决问题的讨论将继续进行下去。

5.3.8.3 如何主持 JRP 会议

JRP 会议以开场白、介绍以及会议议程和会议目标的简短概述开始。JRP 会议主持人将按照准备好的脚本主持会议。为了成功地主持会议，主持人应该遵循以下指南：

- 不要无理由地偏离议程。
- 控制进度（议题分配了明确的时间）。
- 确保记录员能够记录（这可能意味着让用户和管理人员更慢或者更清晰地重述其发言的要点）。
- 避免使用技术行话。
- 应用冲突解决技能。
- 允许充分的休息。
- 鼓励小组取得一致意见。
- 鼓励用户和管理人员参与，不允许个人把持会议。
- 确保与会者遵守制定的会议基本规则。

JRP 会议的一个目标可能是要产生解决问题的想法，实现这个目标的一种方法被称为“头脑风暴”。头脑风暴鼓励参与者在短时间内产生尽可能多的想法而不进行任何分析。

如前所述，JRP 会议的成功在很大程度上依赖计划以及 JRP 会议主持人和记录员的能力，这些技能只有通过培训和经验才能得到提高。JRP 会议通常以一份要求参与者完成的评价调查表作为总结，希望回答将有助于未来 JRP 会议的成功。

JRP 会议的最终产品一般是一份正式的书面文档，这份文档供所有与会者确认在会议期间同意的内容。规格说明的内容和组织形式取决于 JRP 会议的目的，分析员可以根据与会者的角色给不同的人以不同的说明，例如，经理更可能收到一份提供给用户参与者的文档的总结版本（特别是在那些系统所有者很少实际地参与到 JRP 会议中的情况下）。

5.3.8.4 JRP 会议的优点

联合需求计划作为一种调查研究和开发方法具有许多优点。许多公司正开始实现这些优点并正在把 JRP 引入到各自的方法学中。有效组织的 JRP 会议具有以下优点：

- JRP 积极地将用户和管理人员引入到开发项目中（鼓励他们作为项目的主人公）。
- JRP 减少了开发系统所需的时间，这是通过用小组会议代替传统的、耗时的一对一地与每个用户和管理人员面谈实现的。小组会议有助于获得用户和管理人员的一致意见，解决互相矛盾的信息和需求。
- JRP 把原型化技术包括进来作为一种证实需求和获得设计建议批准的手段，此举利用了原型化技术的优点。

JRP 会议的成功取决于 JRP 会议主持人及其计划与主持 JRP 会议的能力。

5.4 调查研究策略

分析员需要使用一种有组织的方法收集事实。一个没有经验的分析员经常直接采用面谈技术。“走到人群中去，那是真正的事实所在！”这样想是完全错误的！这种态度没有认识到一个重要事实：人们必须完成每天的工作！你的工作不是他们的主要责任，你对他们时间的要求对他们来说就是金钱损失。现在你可能在想：“但是我想你曾经说过这个系统是为他们做的，最终用户直接参与到系统开发中是需要的！你不是自相矛盾吗？”

并不是！时间就是金钱。浪费最终用户的时间就是浪费公司的金钱。正是为了把你花的时间大部分都用在最终用户身上，所以才不要直接进行面谈。相反，应该首先收集可以通过其他方法收集到的所有信息。请考虑以下策略：

1. 了解现有文档、表格、报告和文件，你能在没有同任何人接触的情况下了解很多。
2. 如果合适，观察工作中的系统。

3. 根据你已经收集到的所有事实，设计并分发调查表，澄清你没有完全理解的事情。

4. 进行面谈（或小组工作会议）。因为你已经通过较少的用户接触收集了大部分相关事实，所以你现在可以使用面谈来验证和澄清最困难的问题（或者，考虑使用 JRP 技术代替或者补充面谈）。

5. 作为备选，对于没有理解的功能需求或者需要被验证的需求，构造获取原型。

6. 追查到底。使用合适的调查研究技术验证事实（通常是面谈或者观察）。

这个策略并非神圣不可改变。虽然应该为系统开发的每个有关阶段开发一个统一的调查研究策略，但是每个项目都是独特的，所以有时观察和调查表可能是不合适的。但基本思想应该总是在面谈之前收集尽可能多的事实。

复习题

1. 进行需求发现过程的重要性是什么？
2. 如果没有正确而且完全地确定系统需求，可能会有什么后果？
3. 在定义系统需求中有些什么评价标准被认为是关键的？
4. 需求发现过程包括哪些活动？
5. 简要描述鱼骨图的目的和构成要素。
6. 在需求发现阶段通常使用什么技术？这些技术为什么重要？
7. 为什么分析需求是必需的？
8. 从现有文档中收集事实时，系统分析员应该检查哪些文档？
9. 通过在工作环境中观察雇员来收集事实有何缺点？系统分析员应该如何应对这些缺点？
10. 系统分析员可以用来收集信息和观点的调查问卷有哪些类型？
11. 你可以使用哪些方法在交谈中帮助打开话题？
12. 什么是联合需求计划（JRP）？
13. 为什么 JRP 流行？
14. 为什么 JRP 主持人很重要？
15. 在选择 JRP 会议地点时主要关心什么？

问题和练习

1. 你正在管理一个项目。由于项目经费被转移到具有更高优先级的项目，所以项目被延期了两次。系统所有者不想再发生延期，所以他们急于启动这个新系统，并且尽可能快地建造它。他们给你施加了极大的压力，要求你在需求分析上花费不要超过十几天的时间。他们告诉你，如果有什么事遗漏了，可以在以后补上。你真的很想迎合他们，仅有的一点小心谨慎也消失了。匆忙地通过需求分析阶段可能的后果和开销是什么？
2. 当分析问题，系统分析员常犯什么错误？这个错误可能产生的后果是什么？可以使用什么工具帮助避免这个错误？
3. 系统开发人员在每个项目阶段使用调查研究技术。调查研究技术在需求分析阶段比在其他阶段更重要吗？为什么是，或者为什么不是？
4. 系统分析员可以使用什么常用工具和技术记录初始发现？系统分析员应该期望在这时需求完整而正确吗？如果不能，通常的问题是什么？这时项目团队应该重点关注什么？
5. 一旦完成需求分析，产生什么发布物？为什么需要这个发布物？它包括哪些内容？谁是这个发布物的观众或用户，为什么？
6. 你是一个软件开发公司的系统分析员，受雇于一个大型企业做需求分析。在需求发现过程中，你应该收集哪三类现有文档？每类文档都有些什么例子？在收集文档时，系统分析员应该关注什么？
7. 假设你是一个项目中的系统分析员，这个项目涉及修改销售订单过程。由于你的公司每天收到 2500 份左右订单，如果你希望有 95% 的确定性覆盖了所有可能情况，你应该抽样多少份订单？如果每天的销售订单是 25000 份呢？
8. 调研和问卷常被用于收集事实。问卷调查有哪些优点和缺点？什么时候你可以选择自由格式的问卷，而不是固定格式的问卷？确定问卷有效性的一个方法是什么？
9. 为什么使用联合需求计划（JRP）作为调查研究技术？选择用户和管理人员参加 JRP 会议的原则应该是什么？通常由谁来选择他们？主持人和记录员应该具有什么技能？在 JRP 会议期间，IT 职员是什么角色？JRP 会议一般开多长时间？
10. 至少提供 5 个使 JRP 会议成功的关键因素。
11. 当开始需求发现的调查研究部分时，无论有多生气，系统分析员都不应该做的一件事是什么？

项目和研究

1. 系统分析员必须具有问题分析方面的专长。当系统分析员工作时，他们经常发现很难区分症状和问题，确定问题的实际原因。一个可以帮助系统分析员的工具是鱼骨图。
 - a. 寻找一个你的企业、学校或者其他企业正在试图解决的问题，并描述这个问题。
 - b. 按照本章描述的过程创建一张鱼骨图。
 - c. 你在图中从哪个分类开始？在过程中你添加了哪种分类？
 - d. 这张图在寻找问题的实际原因方面有帮助吗？原因是你原先认为的吗，或者是其他不同的原因？
2. 观察工作环境是一种信息时代以前就有的技术，但它仍然十分有效。尽管不是对每种情况都适用，观察人们实际在做什么以及他们如何做在某些情况下要比询问他们更准确！选择一个系统（假设的或者真实的都行）执行以下任务：
 - a. 提供一个系统概述，以及你正试图学习这个系统的哪些内容？
 - b. 使用本章的指南开发一个工作观察计划，格式任意，但一般不要超过 1~2 页。
 - c. 开发一个工作抽样计划，描述你使用的抽样过程。
 - d. 将这个方法与其他调查研究方法比较，你有什么想法？
3. 你是一个系统分析员，正在进行一个项目，为一个拥有几千个雇员、办公室遍布美国的大型企业开发一个内联网。这将是企业的第一个内联网，执行管理层希望它能够有助于提高雇员工作效率和对企业的投入程度。由于企业的规模和地理分布，以及项目的时间约束，没有足够的时间和资源进行个人交谈，所以你决定设计一份调查问卷。
 - a. 你需要收集哪些事实和观点？
 - b. 企业中所有的雇员都需要被调查吗？为什么？如果不需要调查所有雇员，你将如何选择被调查的雇员？
 - c. 你认为什么格式对这份调查问卷最合适？如果是固定格式的，应该使用哪类固定格式的问卷？
 - d. 调查问卷应该多长，以便获得所需信息而不会使雇员不愿意填写？
 - e. 使用本章提供的问题编写指南创建调查问卷。
4. 根据内联网调研的回答，你觉得同其他企业中有开发和/或维护公司内联网经验的人交谈会有帮助吗？
 - a. 你认为在这种情况下哪类会谈最合适——非结构化的还是结构化的？为什么？
 - b. 邀请你的企业的内联网管理员，或者其他企业的，或者学校的，以讨论他们在开发和/或维护内联网方面的经验。
 - c. 使用图 5-3 的格式为例准备一个会谈指南，确保提问没有本章讨论的问题。
 - d. 进行会谈，并记录回答。
 - e. 你觉得在会谈中哪些方面做得好？你下次将在哪些方面进行改进？
5. 肢体语言是交流中十分重要的部分，正如本书中描述的那样。分析员不仅需要知道通过交谈中的肢体语言可以了解到什么，而且要知道他们自己的肢体语言会对交谈过程有什么影响。邀请几个合作者或者同学交流他们希望在内联网上看到的特征；如果可能，选择你了解的交谈者和那些你不熟悉的。按照前面问题中同样的步骤准备会谈。
 - a. 描述你选择的交谈者和你要提问的问题。
 - b. 在每个交谈中，观察交谈者的面部表情。你观察到了什么？面部表情总是同回答一致吗？
 - c. 在每个交谈中，观察交谈者的目光接触。目光接触持续多长时间？观察和描述当你同交谈对象目光接触超过 3 秒到 5 秒钟时间会发生什么。
 - d. 在交谈中试着改变你的空间距离。交谈者做出任何不舒服的表示吗？在哪一点的时候会表现出来？
 - e. 你注意到你熟悉的人和你不熟悉的人之间在肢体语言上有什么不同吗？
 - f. 在诱发信息方面你所做的什么是最成功的，什么是最不成功的？
6. 分析员在项目的需求发现阶段一般会接触到机密的或者敏感的数据，特别是在调查研究中。分析员需要知道什么会违背职业道德，是否通过了委托或者没有，以及可能的后果。在网上或者你的学校图书馆的商业期刊上查找关于违背职业道德的事件的文章。
 - a. 你找到了哪些文章？
 - b. 这些事件有什么特征？
 - c. 后果是什么？
 - d. 在每个事件中，分析员的个人责任是什么？
 - e. 在企业或个人的层面如何来防止这些事件或降低其严重程度？

小型案例

1. 在第4章中，你为项目开发了可行性研究。经济可行性评估明显地受到无形资产的影响，它们的价值部分地来自交谈和问卷调查。开发面谈问题以确定雇员远程办公的价值。
 - a. 从提供给一组雇员的非结构化问题开始决定什么事情影响了雇员，以及他们如何看待远程办公。
 - b. 一旦你知道了什么问题围绕着雇员对远程办公的理解，以及为什么他们可能喜欢/不喜欢它，创建关于那些问题的开放的但结构化的问题，并同另一组雇员交流。为什么我们使用两个不同组的雇员完成这个过程？
2. 基于在前一题中你的发现，为大量雇员开发一个问卷调查。为什么使用一种匿名调查完成这个分析？
3. 你负责为你的学校开发一个新的联机课程注册系统。开发一套面谈问题以确定学生、注册职员和办公人员对联机注册系统的问题和需要。
4. 讨论有偏见的或者具有导向性的问题可能对分析的影响。创建一个没有偏见的面谈问题，和一个有偏见的或具有导向性的问题。把这些问题发给5个人。你得到哪种回答？是你想要的回答吗？

团队和个人练习

1. 创建一套有偏见的、具有导向性的或者含沙射影的面谈问题。把它们发给课堂上的另一位学生。请这个学生不用回答问题，而是告诉你你有什么偏见，你正在寻找什么样的回答。
2. 课堂练习：创建一套没有偏见的面谈问题，可以是某个特别的主题。把问题发给班级。但是，套上封套、大头针等，这会引导班上同学按照某种特定方式回答。很有趣，用视觉辅助、小道具等进行试验。
3. 在过去的调查研究中已经发现被允许远程办公的雇员实际上每周大约超额工作3个小时。但远程办公常常被雇主用作协商工具——为了远程办公，雇员必须接受较低的工资，一般低10%。你对此有何感想？

使用用例建模系统需求

本章概述和学习目标

本章讲述采用用例模型记录系统需求的工具和技术。捕捉和记录系统需求是信息系统开发项目成功的关键。以用户可理解的方式从用户的观点记录需求促进了用户的参与，将极大地提高项目成功的可能性。本章将讲述以下需求用例建模内容：

- 描述用例建模的优点。
- 定义参与者和用例，并能够从上下文图以及其他资源中确定参考图和用例。
- 描述四类参与者。
- 描述用例模型图中可能出现的关系。
- 描述准备用例模型图的步骤。
- 描述如何构造用例模型图。
- 描述用例描述的各节内容。
- 定义用例分级的目的、优先权矩阵，以及用例依赖关系图。

本章关键术语

以用户为中心的开发（user-centered development）是一个系统开发过程，该过程基于对关联人员的需求，以及对开发该系统原因的充分理解之上。

用例建模（use-case modeling）是使用业务事件、发起业务事件的人，以及系统如何响应这些事件来建模系统功能的过程。

用例图（use-case diagram）是描述系统与外部其他系统以及用户之间交互的图形。换句话说，用例图描述了谁将使用系统，用户希望以什么方式与系统交互。

功能分解（functional decomposition）是将一个系统拆分成子构件的活动。

用例描述（use-case narrative）是业务事件以及用户如何同系统交互以完成任务的文字描述。

用例（use case）是一个行为上相关的步骤序列（一个场景），既可以是自动的也可以是手工的，其目的是完成一个单一的业务任务。

参与者（actor）代表了需要同系统交互以交换信息的任何事物。

时序事件（temporal event）是由时间触发的系统事件。

关联关系（association）是一个参与者与一个用例发生交互的关系。

扩展用例（extension use case）是一个由从某个更复杂的用例中提取出来的步骤构成的用例，以便简化原始用例并扩展其功能。

抽象用例（abstract use case）通过组合几个用例中公共的步骤来降低用例之间的冗余。

依赖（depends on）是用例之间的一种关系，表示一个用例需要等到另一个用例执行之后才能执行。

继承（inheritance）是参与者之间的一种关系，创建继承关系的目的是当一个抽象参与者继承多个实际参与者的角色时简化绘图。

业务需求用例（business requirements use case）是在需求分析过程中为了捕捉用户与系统之间交互而建立的用例，并没有技术和实现细节，也称为基本用例。

用例分级和评估矩阵（use-case ranking and priority matrix）是用来评估用例决定其优先级的工具。

用例依赖关系图（use-case dependency diagram）是用例之间的依赖关系的图形化表述。

6.1 用例建模简介

对于信息系统开发团队（尤其是系统分析员）来说，最主要的挑战就是能够从关联人员那里提取出正确的确实需要的系统需求，并以关联人员可以理解的方式进行说明，以便需求可以得到证实和验证。

信息技术团体在向用户说明需求（特别是功能需求）方面总是很难。过去，我们使用数据模型、过程模型、原型系统以及需求规格说明等工具，我们理解而且熟悉这些工具，但那些没有受过软件开发实践教育的用户很难理解。因此，许多开发项目一直苦于范围蔓延、费用超支和进度蔓延问题。经常系统开发出来并部署后，却并不能真正满足用户的需要。有些项目被束之高阁而根本就没有用，更多的项目在开发结束之前就被取消。著名的调查公司 Standish Group，分别于 1994 年、1996 年和 1998 年研究了 23 000 个 IT 项目^①。如图 6-1 所示，1998 年的研究发现超过四分之一的项目成功（在预算内、按时并实现了所有功能），超过四分之一的项目失败（在完成之前被取消），不足一半的项目被认为是“有问题的”——项目结束并投入运行，但或者是超过了预算，或者是超过了时间，或者是没有实现用户要求的所有功能。研究中反映良好的一面是我们用来开发信息系统的手段得到了提高，软件开发工业已经懂得了：为了成功地计划、分析、设计、构造和部署一个信息系统，系统分析员首先必须理解关联人员的需求，以及开发该系统的原因——以用户为中心的开发。通过关注系统的用户，分析员能够把重点放在系统如何使用，而不是系统如何构造上。用例建模是一种促进以使用为中心的开发方法。

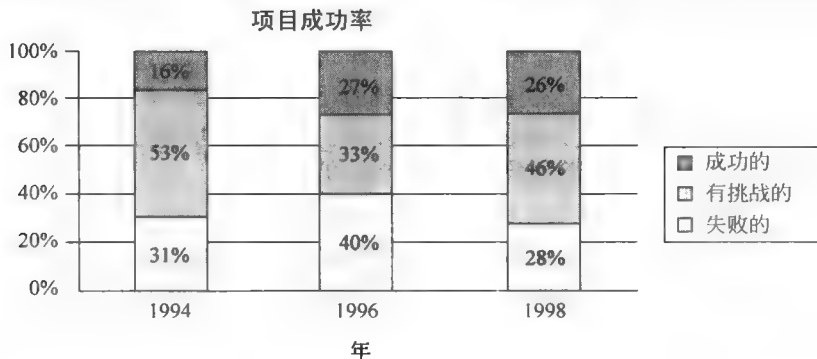


图 6-1 Standish Group 报告的项目成功率

用例建模最初由 Ivar Jacobson 博士于 1986 年发明，在 1992 年他发表《Object-Oriented Software Engineering》一书后流行起来。Jacobson 博士使用用例建模作为他的 Objectory 方法学的框架，该方法学成功地用于开发面向对象的信息系统。在从用户和关联人员那里确定系统需要做什么方面，用例建模很有用。如今，用例建模被广泛地认同为定义、记录和理解信息系统功能需求的最佳实践。

6.2 用例建模的系统概念

用例建模主要有两个产物。第一个是用例图，它以图形化的方式将系统描述成用例、参与者（用户）及其之间的关系。用例图在高层交流了系统必须处理的业务事件的范围。图 6-2 是一个用例图的例子。用例图中显示了每个系统功能，或者业务事件（用椭圆表示）和参与者，或者系

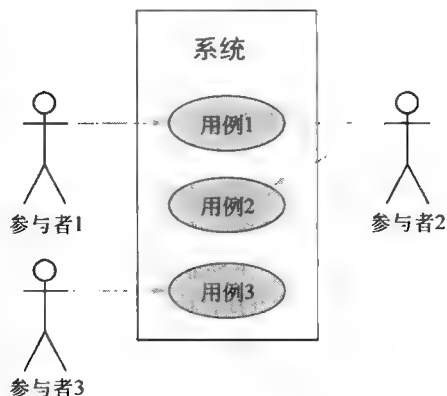


图 6-2 用例模型图示例

① The Standish Group International, Inc., “CHAOS: A Recipe for Success” (电子版), 1999. Retrieved December 5, 2002, from www.pm2go.com/sample_research/chaos1998.pdf. The Standish Group 因为在 IT 界首先研究和分析依赖关系而著名。

统用户，他同那些功能交互。如图 6-2 所示，参与者可以放在用例图的任何一边，并且可以同一个或多个用例交互。用例图很简单，但它开始了一个重要的过程，这个过程称为功能分解，将一个系统拆分成它的子构件的活动。立刻理解整个系统是不可能的，但理解并描述系统的每个部分是可能的。第 2 个产物——用例描述，填充了每个业务事件，并说明了用户如何同系统交互的细节。用例说明将在本章后面详细介绍。

6.2.1 用例

用例建模通过使用用例工具确定和描述系统功能（参见 4.5.1 节）。用例从外部用户的观点并以他们可以理解的方式和词汇描述系统功能。为了正确全面地达到这个目标，需要用户的高度参与，并需要有熟悉业务过程或业务事件的主题领域专家。

用例用一个水平的椭圆（用例名称在椭圆上面、下面或内部显示）表示，如右图所示。一个用例代表了系统的一个单一的目标，描述为实现此目标的活动和用户交互的一个序列。用例是一种理解和记录系统需求的出色的技术。一个用例本身并不是一个功能需求，但用例所讲述的故事（场景）包含了一个或者多个需求。



用例最初在系统生命周期的需求阶段定义，并在整个生命周期中不断地细化。在需求获取阶段，用例用来捕捉业务问题的本质，建模建议的系统的（高层）功能。另外，它们是确定系统的数据实体（见第 7 章）或对象的起点。在需求分析阶段，用例被细化，用来更详细地建模系统的使用（见第 9 章）。换句话说，用例被修改以说明用户试图实现什么以及为什么实现。在设计阶段，用例被细化，用来建模用户如何实际使用系统，考虑界面和系统约束条件（见第 17 章）。这些类型的用例辅助确定对象或系统行为，设计界面和代码说明，并作为测试系统的计划。在构造阶段，用例辅助开发人员进行编程和测试。这些用例也作为准备用户和系统文档的基础，以及用户培训的工具。另外，因为用例包含了大量的系统功能细节，它们将是验证系统的一个稳定的资源。

6.2.2 参与者

发起或者触发用例的外部用户称为参与者。为了完成某些具有可度量值的业务任务，参与者发起系统活动——用例。我们以大学学生注册秋季课程为例进行说明。参与者是学生，业务事件（或用例）是注册课程。参与者代表了同系统交互的用户实现的角色，而不代表一个人或者工作职位。事实上，参与者不必是一个人，它也可以是一个组织、另一个信息系统、一个外部设备（如热传感器），或者甚至是时间概念（将在后面讨论）。参与者的图形表示是一个小人，参与者的名称标记在下面。

注意有四类主要的参与者：

- 主要业务参与者——主要从用例执行中获得好处的关联人员。主要业务参与者可能（也可能不能）发起业务事件。例如，在顾客每周五从支付系统收到发票（某种可度量的价值）这个业务事件中，顾客并没有发起事件，但他是某些价值的主要接收者。
- 主要系统参与者——直接同系统交互发起或触发业务或系统事件的关联人员。主要系统参与者可能会同主要业务参与者交互，以便使用系统。他们通过直接使用系统发起事件，为主要业务参与者提供价值。例如，零售店店员把商品目录给顾客，银行出纳处理一个银行事务。对于业务参与者直接同系统交互的事件，主要业务参与者和主要系统参与者可能是同一个人——例如，通过 Web 站点预订旅店的人。
- 外部服务参与者——响应来自用例的请求的关联人员（例如，信用卡部门认证一个信用卡支付）。
- 外部接收参与者——不是主要参与者，但从用例接收某些可度量的或可观察的价值（输出）的关联人员（例如，当顾客下了一个订单后，仓库收到一个打包单准备发货）。



参与者符号

在许多信息系统中，业务事件由日期或时间触发。请考虑以下的例子：

- 一个信用卡公司的收账系统在每个月的第 5 天（开票日）自动地生成账单。
- 一个银行在每天下午 5 点对账。

- 每天晚上，系统自动生成一份报告列出哪门课程已经停止注册（没有座位了），哪门课程仍可以注册。

这些事件都是**时序事件**的例子。这些例子中的参与者是谁？上面列出的所有例子事件都是当到了某个确定的日期或者时间时自动执行的（或者自动触发的）。我们说一个时序事件的参与者是时间。

6.2.3 关系

关系在用例图上用两个符号之间的一条线表示。根据该线如何绘制以及连接何种类型的符号，关系的含义可能不同。在以下小节中，我们将定义用例图中的不同关系。

6.2.3.1 关联关系

只要用例描述了参与者之间的交互，一个参与者和一个用例之间就存在一个关系。这个关系称为**关联关系**。如图6-3所示，关联关系是用一条连接参与者和用例之间的实现表示。关联关系包含一个箭头，一头为用例另一头为参与者（①），表示参与者发起用例。没有箭头的关联关系（②）表示用例与外部服务者或接收者参与者交互。当一个参与者与一个用例相关联时，我们称参与者同用例通信。关联关系可以是双向的，也可以是单向的。

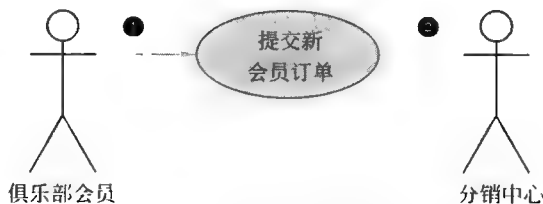


图6-3 关联关系的例子

6.2.3.2 扩展关系

一个用例可能包含由多个步骤组成的复杂功能，使得用例难以理解。为了简化用例，使其更容易理解，我们可以将较复杂的步骤提取成专门的用户。这样得到的用例称为**扩展用例**，它扩展了原始用例的功能。扩展用例和它扩展的用例之间的关系称为**扩展关系**。一个用例可以有多个扩展关系，但一个扩展用例只能被它扩展的用例调用。如图6-4所示，扩展关系用一条箭头线表示（实线或者虚线），起点是扩展用例，终点是被扩展的用例。每个扩展关系线标记“<<extends>>”。一般来说，扩展关系用例不在需求阶段确定，而在分析阶段确定。

6.2.3.3 使用（或包含）关系

你可能经常会发现两个或多个用例执行同样的功能步骤。最好把这些公共步骤提取成独立的用户，称为**抽象用例**。抽象用例代表了某种形式的“复用”，是降低用例之间冗余的极好的工具。抽象用例可以被另一个需要使用它的功能用例访问。抽象用例和使用它的用例之间的关系称为**使用关系**（某些用例模型称为包含关系）。在图6-5中，使用关系表示成一个箭头线（实线或虚线），起点是原始用例，指向它使用的用例。每个使用关系线标记“<<uses>>”。一般来说，抽象用例不在需求阶段确定，而在分析阶段确定。

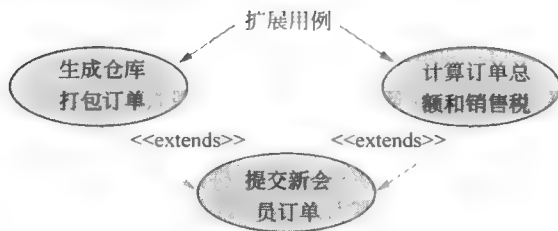


图6-4 扩展关系的例子

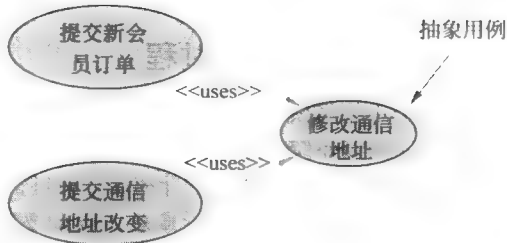


图6-5 使用关系的例子

6.2.3.4 依赖关系

作为项目经理或者主要开发人员，了解哪个用例与其他用例有依赖关系很重要，这样可以决定用例开发的顺序。以银行业务为例，“取款”用例需要等到“存款”用例执行之后才能运行，而“存款”用例要在“建立账户”用例之后运行。因为这些依赖关系，开发团队最可能首先开发“建立账户”用例，然后“存款”用例，最后“取款”用例。用例图使用**依赖关系**建模系统的用例之间的依赖性，为

规划和调度提供了一个极好的模型。依赖关系如图 6-6 所示, 用一个箭头线表示 (实线或虚线), 起点是某个用例, 指向它依赖的用例。依赖关系线标记 “<<depends on>>”。

6.2.3.5 继承关系

当多个参与者共享同样的行为时 (即他们可以发起同样的用例), 最好将这些公共行为分配给一个新的抽象参与者, 以便降低与系统通信的冗余。例如, 一个图书馆的资助人是正式成员, 他有权从图书馆 “查询库存”, 也可以 “借书”。由于许多图书馆都是公共设施, 它们欢迎访客使用它们的服务, 例如 “查询库存”, 但访客不享受扩展服务 (例如 “借书”)。通过创建一个称为客户的抽象参与者, 资助人和访客从他继承, 我们只需建模一次发起用例 “查询库存” 的关系。在用例图中, 继承关系表示成图 6-7 中的箭头线。

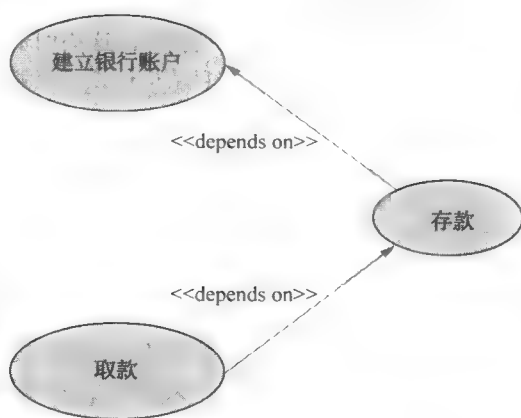


图 6-6 依赖关系的例子

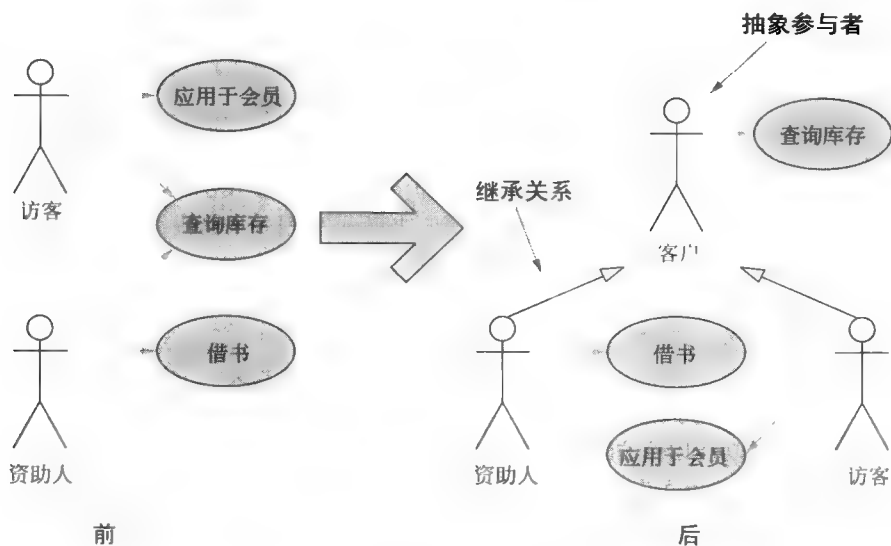


图 6-7 继承关系的例子

6.3 需求用例建模过程

构造需求用例模型的目的是提取和分析足够的需求信息, 准备一个模型, 该模型表述了用户需要什么, 而不涉及系统如何将如何构造和实现的特定细节。遵循这个方法最终将得到一个更健壮不易受变化影响的设计。但为了有效地估计和调度项目, 模型可能需要包含初始的 “系统实现假设”。当准备这个模型时, 分析员不要陷入分析瘫痪症很重要。速度是关键。在这个阶段, 并非需要了解所有的事实, 但通过迭代和增量开发, 方法学使得在项目以后增加新的需求不会严重影响最终方案的部署。下面我们详细介绍产生需求用例模型的几个步骤。

6.3.1 第 1 步: 确定业务参与者

为什么首先要确定参与者? 通过关注参与者, 我们可以把重点放在如何使用系统, 而不是如何构造系统上。关注参与者有助于提炼和进一步定义系统的范围和边界。参与者也决定了系统需求的完整性^①。

① Frank Armour and Granville Miller, *Advance Use Case Modeling* (Boston: Addison-Wesley, 2001)。

首先确定参与者的一个好处是可以确定日后进行面谈和观察以完善用例模型的候选人。而且，当完成用例模型后，这些参与者可以验证用例。

在哪里寻找潜在的参与者呢？以下的参考资料是极好的信息源：

- 标示系统范围和边界的上下文图。
- 现有系统的文档和用户手册。
- 项目会议和研讨会的记录。
- 现有的需求文档、项目章程或工作陈述。

当寻找参与者时，提出以下问题：

- 谁或者什么为系统提供输入？
- 谁或者什么接收系统的输出？
- 需要与其他系统连接的接口吗？
- 是否存在在预定的时间自动触发的事件？
- 谁将维护系统中的信息？

应该使用名词或名词词组命名来命名参与者。

当确定参与者时，从用户的角度出发并使用用户的词汇给出参与者的文字定义。图 6-8 是一个参与者词汇表的模板，可以用来记录参与者。这个例子包含了音阶娱乐俱乐部会员服务系统的参与者的部分清单。

参与者词汇表

词 汇	同义词	描 述
1. 潜在的会员		提交加入俱乐部的订阅订单的个人或者公司
2. 俱乐部会员	会员	已经加入俱乐部的个人或者公司
3. 过去的会员	非活动会员	一类会员，他们曾经有过合同关系，但在最近的 6 个月内没有下过订单，但仍保持良好的身份记录
4. 市场部		响应创建折扣和订阅程序并为公司进行销售的组织部门
5. 会员服务部		按照合同为音阶娱乐俱乐部的顾客提供联系服务的组织部门
6. 分销中心	仓库	存储和维护音阶娱乐俱乐部产品库存并处理顾客发货和退货的实体
7. 应收账款部门		处理顾客付款和收费以及维护顾客账户信息的组织部门
8. 时间		触发时序事件的参与者

图 6-8 音阶娱乐俱乐部会员服务系统的参与者的部分清单

6.3.2 第 2 步：确定业务需求用例

一个典型的信息系统可能包含许多用例。在需求分析阶段，出于时间和经费的考虑，我们仅仅粗略地确定和记录了最关键、最复杂和最重要的用例，它们经常被称为基本用例。一个业务需求用例捕捉了与用户的交互，并没有技术和实现细节。既然用例描述了真实世界中参与者如何与系统交互，那么寻找业务需求用例的一个好方法就是检查参与者以及他们如何使用系统。当寻找用例时，询问以下问题：

- 参与者的主要任务是什么？
- 参与者需要系统什么信息？
- 参与者为系统提供什么信息？
- 系统需要通知参与者发生的变化和事件吗？
- 参与者需要通知系统发生的变化和事件吗？

上下文图是分析参与者和发现潜在用例的极好来源。上下文图已在第 4 章讨论了。它们来源于传统的过程建模（第 8 章），但甚至在采用面向对象方法的项目中也有用。让我们审视音阶娱乐俱乐部会员服务系统的上下文图（见图 6-9）。我们可以通过看图、确定系统的主要输入输出，以及提交和接收输入输出的外部各方来确定潜在的用例。触发组织内的业务事件（如提交会员订单）的主要输入被认为是用例，提供这些输入的外部各方被认为是参与者（如俱乐部会员）。注意那些系统请求的结果并不被看作是输入，（例如，信用卡公司响应认证请求，如图 6-9 所示）应收账款部门参与者响应会员信用状态信息。

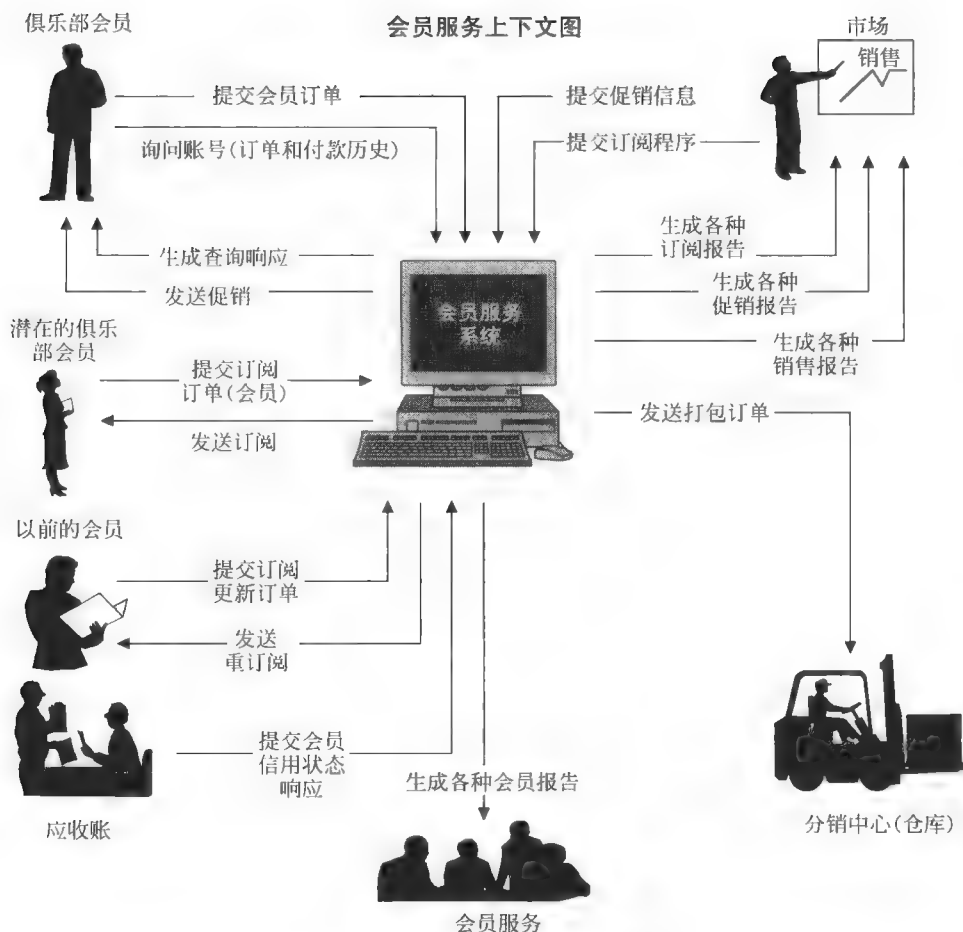


图 6-9 会员服务系统的上下文图

用例使用输入的名称前加一个行动动词命名。例如，如果订阅订单是输入，用例的一个恰当的名称是提交订阅订单。时序事件用例通常是分析系统关键输出的结果。例如，基于时间或日期产生的输出（如月报表或年报表）被看作是用例，其参与者是时间。在图 6-9 中，假设会员服务部门收到的一种报告是 10-30-60 天默认合同报告，该报告每天自动生成。既然报告的生成由时间触发，所以需要有一个用例处理该事件，我们将其命名为每日生成 10-30-60 天默认合同报告。注意单独的报告经常不在上下文图中列出，因为它们太多了，会使得图形杂乱难以阅读。调查相应的关联人员收到的输出的类型及其特征（数量、频率和触发机制）以确定“隐藏的用例”是系统分析员的责任。

图 6-10 是一个用来记录用例的用例词汇表的模板。这个例子包含了音阶娱乐俱乐部会员服务系统的用例、从上下文图和其他来源确定的参与者的部分清单。

6.3.3 第 3 步：构造用例模型图

一旦确定了参与者和用例，就可以使用用例模型图描述系统范围和边界。图 6-11 显示了图 6-10 的用例的用例模型图，它使用 Popkin Software 的 System Architect 构造，表示了参与者和用例之间的关系。另外，用例被组合成业务子系统。子系统（UML 包符号）表示业务过程的逻辑功能区。将系统行为分解成子系统对于理解系统结构很重要，也是定义开发策略的关键——哪个用例先开发，由谁开发。我们已经标出了参与者和用例“发起”之间的关系，因为工具不支持双向箭头线。因为图形空间的限制，我们也没有包括外部服务者和接收参与者。为了建模一个系统的所有用例，可能需要创建多个用例模型图——系统可能包括很多用例。在这种情况下，你可能会为每个子系统创建一个独立的用例模型图。

用例词汇表

用例名称	用例描述	参与者
提交订阅订单	该用例描述一个潜在的会员通过订阅加入俱乐部 (“用1美分的价格购买12张CD并答应在两年内按照正常价格购买4张以上的CD”)	潜在的会员 (主要业务) 分销中心 (外部接收者)
提交订阅更新订单	该用例描述一个过去的会员通过订阅加入俱乐部 (“用1美分的价格购买12张CD并答应在两年内按照正常价格购买4张以上的CD”)	过去的会员 (主要业务) 分销中心 (外部接收者)
提交会员资料变更	该用例描述一个俱乐部会员修改他的个人资料, 例如: 邮寄地址、电子邮件地址、密码和订单偏好	俱乐部会员 (主要业务)
下新订单	该用例描述一个俱乐部会员提交一个音阶娱乐俱乐部产品的订单	俱乐部会员 (主要业务) 分销中心 (外部接收者) 应付账/应收账 (外部服务者)
修改订单	该用例描述一个俱乐部会员修改一个以前下的订单的事件 (订单必须还没有履行)	俱乐部会员 (主要业务) 分销中心 (外部接收者) 应付账/应收账 (外部服务者)
取消订单	该用例描述一个俱乐部会员取消一个以前下的订单的事件 (订单必须还没有履行)	俱乐部会员 (主要业务) 分销中心 (外部接收者) 应付账/应收账 (外部服务者)
产品查询	该用例描述一个俱乐部会员为可能的购买查阅产品的事件 (由 Web 访问需求驱动)	俱乐部会员 (主要业务)
购买历史查询	该用例描述一个俱乐部会员查阅他的购买历史 (三年内)	俱乐部会员 (主要业务)
建立新会员订阅程序	该用例描述市场部建立一个新的会员订阅计划来吸引新的会员	市场部 (主要业务)
提交订阅程序变更	该用例描述市场部为俱乐部会员修改订阅计划 (例如, 延长履行周期)	市场部 (主要业务)
建立过去会员重订阅读程序	该用例描述市场部为建立一个重订阅读计划以吸引回以前的会员	市场部 (主要业务)
提交会员资料变更	该用例描述市场部建立一个新的促销计划以吸引活动的和非活动的会员订购产品。注意: 促销产品以专门的标题名称 (通常是新的) 来表示, 公司试图以一个特殊价格销售它。这些促销产品被集成到一个目录中并发送 (或者沟通) 给所有会员	市场部 (主要业务)
修改促销	该用例描述市场部修改促销的事件	市场部 (主要业务)
每日生成 10-30-60 天默认合同报告	该用例描述每天生成一个报告, 列出还没有通过购买一定数量的产品履行合同的会员。该报告按照会员 10 天前过期、30 天前过期、60 天前过期排序	时间 (发起参与者) 会员服务 (主要 ^❶ ——外部接收者)

❶ 之所以被认为是主要的, 因为它收到了某些可度量的价值。

图 6-10 音阶娱乐俱乐部会员服务系统的用例的部分清单

6.3.4 第4步: 记录业务需求用例描述

当准备用例描述时, 首先在高层记录, 以便尽快理解系统的事件和量级。然后再回到每个用例, 扩展它以完全地记录业务需求描述。图 6-12 表示了会员服务系统的下新订单用例的需求用例描述。注意它精炼地描述了事件, 包括以下内容:

- ❶ 作者——编写用例, 并为需要该用例额外信息的人提供联系人姓名。
- ❷ 日期——用例修改的日期。
- ❸ 版本——用例的当前版本 (如 1.0)。
- ❹ 用例名称——用例名称应该描述用例试图实现的目标。名称应该以动词开始 (例如, 输入新会员订单)。

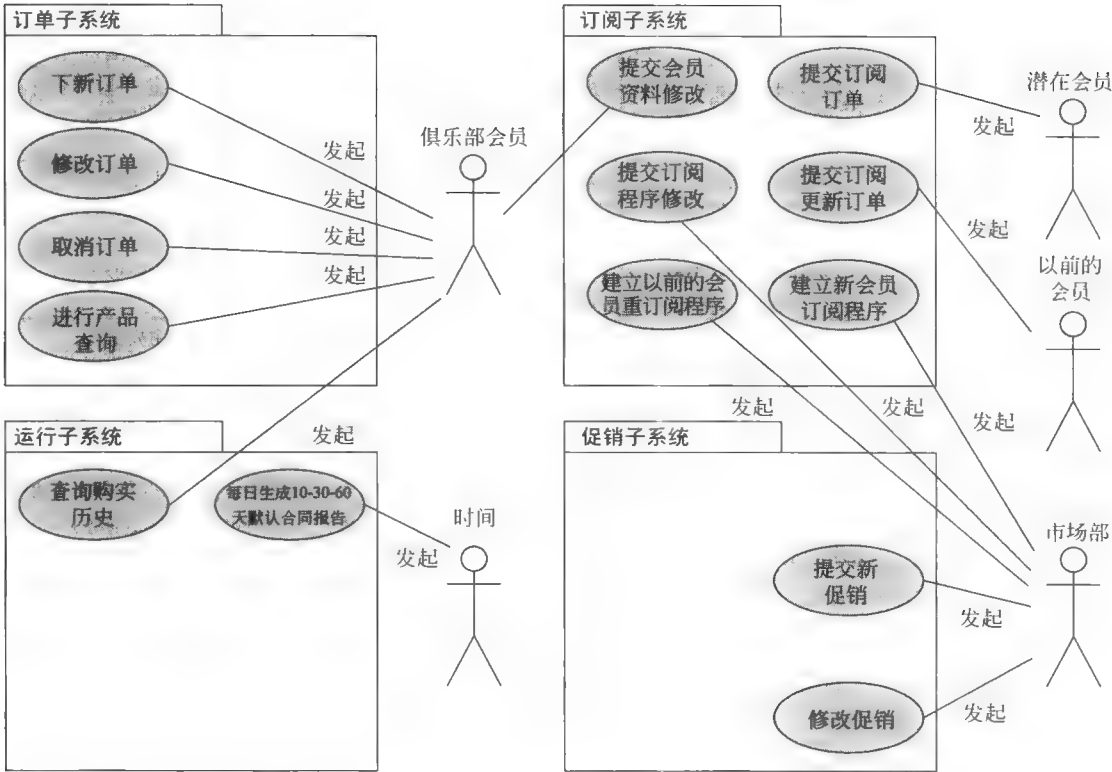


图 6-11 音阶娱乐俱乐部会员服务系统的用例模型图

- ① 用例类型——在进行用例建模时，首先构造业务需求用例（关注各种关联人员的战略目标）。这类用例是面向业务的，反映了系统期望行为的高层视图。它没有技术细节，可以包含人工操作，以及将被自动化的活动。业务需求用例提供了对问题领域和范围的一般性理解，但不包含与开发人员交流系统应该做什么所需的细节。
- ② 用例 ID——唯一标示用例的标识符。
- ③ 优先权——优先权表达了用例的重要性（高、中、低）。
- ④ 来源——来源定义了触发用例创建的实体。这可能是一个需求，一个特殊的文档或者某个关联人员。
- ⑤ 主要业务参与者——主要业务参与者是通过获得某些可度量或可观察的价值从用例的执行中获益的关联人员。
- ⑥ 其他参与者——参与用例的其他参与者包括：发起参与者、主持参与者、服务/接收参与者、附属参与者。总是包括参与者参与的方式。
- ⑦ 有利益的关联人员——关联人员是对软件系统的开发和运行有切身关系的人。有兴趣的关联人员是对用例的目标有一定利益的人（或者参与者）。
- ⑧ 描述——一段简单的总结性描述，概要描述用例及其活动的目的。

记录用例的事件过程

对于每个确定的高层事件，我们现在需要扩展用例的典型事件过程和替代过程。用例的典型事件过程是从参与者发起用例直到业务事件结束的步骤描述。在本节中，我们仅仅包含了主要时间发生的主要步骤（典型过程）。替代过程记录了用例的异常或例外分支。图 6-13 表示了会员服务系统的下新订单用例的需求用例描述。注意图中包含了以下内容：

- ① 前置条件——前置条件是在用例执行之前，关于系统状态的约束条件。一般是指需要首先执行的另一个用例。

会员服务系统

作者： ①

日期： ②

版本： ③

用例名称	下新订单 ①	用例类型：
用例 ID	MSS-BUC002.00 ④	业务需求： <input checked="" type="checkbox"/>
优先级	高 ⑤	⑥
来源	需求——MSS-R1.00 ⑦	
主要业务参与者	俱乐部会员 ⑧	
其他参与者	• 仓库（外部接收者） ⑩ • 应付账/应收账（外部服务者）	
其他有兴趣的关联人员	• 市场部——对销售活动感兴趣，为了计划新的促销 ⑪ • 采购部——对销售活动感兴趣，为了补充库存 • 管理层——对销售活动感兴趣，为了评估公司性能和顾客满意度	
描述 ⑨	该用例描述一个俱乐部会员提交一个音阶娱乐俱乐部产品的订单。会员的资料信息以及他的账号被验证。一旦验证产品有库存，就给仓库发出一个发货订单准备发货。对于没有库存的产品，生成一个退单。一旦完成，会员将得到一份订单证实	

图 6-12 下新订单用例描述的高层版本

- ② 触发器——触发器是发起用例执行的事件。这通常是一个动作，例如一个客户走到一个销售柜台，或者一张支票通过邮寄寄到了。时间也可以触发用例。
- ③ 典型事件过程——典型事件过程是参与者和系统为了满足用例目标执行的常规活动序列，包括：系统与参与者的交互和为响应交互系统执行的活动。注意角色的动作记录在左列，系统的动作记录在右列。
- ④ 替代过程——替代过程记录如果典型过程出现异常或变化时的用例行为。当用例中存在决策点或者异常时，就会存在替代过程，需要典型过程以外的额外步骤。
- ⑤ 结论——结论描述用例什么时候成功结束，换句话说，什么时候主要参与者收到某些可度量的价值。
- ⑥ 后置条件——后置条件是在用例执行之后，关于系统状态的约束条件。这可以是记录在数据库中的数据或者发送到客户的一张发票。
- ⑦ 业务规则——业务规则说明新系统必须服从的业务政策或规程。这可以包括运输货物的计算或者授权信用的规则。
- ⑧ 实现约束和说明——实现约束和说明表述可能影响用例实现和有助于架构规划的非功能需求，可以包含的项目有：出现频率、数据量、接口需求等。
- ⑨ 假设——记录用例时作者做出的假设。
- ⑩ 开放问题——在用例最终确定之前需要解决或研究的问题。

业务需求用例是出色的工具，它们描述了组织必须处理和响应的事件，但缺少关于接口和需要信息技术自动化活动的信息。第9章将讲解如何扩展用例以包含技术和实现细节。

6.4 用例和项目管理

用例建模的一个优点是用例模型可以用来驱动整个系统开发工作。一旦完成了业务需求用例模型，项目经理或者系统分析员就用业务需求用例计划（估计和安排进度）项目的构建周期。当应用迭代和增量方法进行软件开发时这一点就特别重要。一个构建周期包括系统分析、设计和构造活动，根据用例的重要性和构造时间进行框定。换句话说，在每个构建周期将开发多个用例，最初实现一个简化的版本，之后在以后的周期中实现完全的版本。为了决定用例的重要性，项目经理或者系统分析员将填写用例分级和评估矩阵，并使用来自关联人员和开发团队的输入构造用例依赖关系图。下面的小节将讲述如何使用这些工具。

会员服务系统

作者: _____

日期: _____

版本: _____

用例名称	下新订单	用例类型:
用例 ID	MSS-UC002.00	业务需求: <input checked="" type="checkbox"/>
优先级	高	
来源	需求——MSS-R1.00	
主要业务参与者	俱乐部会员	
其他参与者	<ul style="list-style-type: none"> 仓库 (外部接收者) 应收账款 (外部服务者) 	
其他有兴趣的关联人员	<ul style="list-style-type: none"> 市场部——对销售活动感兴趣, 为了计划新的促销 采购部——对销售活动感兴趣, 为了补充库存 管理层——对销售活动感兴趣, 为了评估公司性能和客户满意度 	
描述	该用例描述一个俱乐部会员提交一个音阶娱乐俱乐部产品的订单。会员的资料信息以及他的账号被验证。一旦验证产品有库存, 就给仓库发出一个发货订单准备发货。对于没有库存的产品, 生成一个退单。一旦完成, 会员将得到一份订单证实。	
前置条件	提交订单的一方 (个人或者公司) 必须是会员。	
触发器	当新订单提交时, 用例被触发。	
典型事件过程	<p>参与者动作</p> <p>第 1 步: 俱乐部会员提供他的资料信息以及订单和支付信息。</p>	<p>系统响应</p> <p>第 2 步: 系统验证所需的所有信息都提供了之后做出响应。</p> <p>第 3 步: 系统根据以前记录的资料验证俱乐部会员的资料信息。</p> <p>第 4 步: 对于订购的每个产品, 系统验证产品标识。</p> <p>第 5 步: 对于订购的每个产品, 系统验证产品可用性。</p> <p>第 6 步: 对于每个可用的产品, 系统决定向俱乐部会员收取的价格。</p> <p>第 7 步: 一旦处理了所有订购的产品, 系统决定订单的总价格。</p> <p>第 8 步: 系统检查俱乐部会员账号的状态。</p> <p>第 9 步: 系统验证俱乐部会员的支付 (如果提供了)。</p> <p>第 10 步: 系统记录订单信息, 然后将订单发送到相应的分销中心 (仓库) 履行。</p> <p>第 11 步: 一旦处理完订单, 系统生成一个订单确认, 并发送给俱乐部会员。</p>
替代事件过程	<p>替代第 2 步: 俱乐部会员没有提供处理订单所需的所有信息, 通知俱乐部会员并提示重新提交。</p> <p>替代第 3 步: 如果提供的俱乐部会员信息与原先记录不同, 则验证哪个记录是最新的, 然后相应地修改俱乐部会员信息。</p> <p>替代第 4 步: 如果俱乐部会员提供的产品信息与音阶俱乐部的任何产品都不匹配, 则通知俱乐部会员并要求澄清。</p> <p>替代第 5 步: 如果无法提供订购的数量, 则生成一个延迟交货单。</p> <p>替代第 8 步: 如果俱乐部会员的账号状态不良, 记录订单信息, 把它设为挂起状态。通知俱乐部会员其账号状态, 以及订单被挂起的原因。终止用例。</p> <p>替代第 9 步: 如果俱乐部会员提供的付款 (信用卡) 不能通过验证, 通知俱乐部会员, 并要求另一种支付方式。如果俱乐部会员不能提供另一种支付方式, 则取消订单并终止用例。</p>	
结论	当俱乐部会员收到订单确认时, 该用例结束。	
后置条件	订单被记录下来, 如果订购的产品有货, 将发货。对于缺货的产品, 生成一个延迟交货单。	
业务规则	<ul style="list-style-type: none"> 响应促销的俱乐部会员或者使用信用卡的会员可能会影响每个订购项目的价格。 现金或支票不与订单一起接收。如果提供, 将被退回俱乐部会员。 只有当产品发货时, 才向俱乐部会员收费。 	
实现约束和说明	<ul style="list-style-type: none"> 要为会员服务提供 GUI, 为俱乐部会员提供 Web 界面。 	
假设	将在日报中通知采购部门延迟交货单 (独立的用例)。	
环境	需要决定如何指定分销中心。	

图 6-13 下新订单用例描述的扩展版本

6.4.1 分级和评估用例

在大多数软件开发项目中，首先开发最重要的用例。为了决定用例的优先级，项目经理使用称为**用例分级和评估矩阵**的工具。这个矩阵使用来自关联人员和开发团队的输入填写。该矩阵（引用自 Craig Larman^①）使用6个标准按1~5级评估用例。6个标准是：

1. 对架构设计的重要影响。
2. 容易实现但包含重要功能。
3. 包含有风险、时间紧迫或者复杂的功能。
4. 需要大量的研究或者新的、有风险的技术。
5. 包含主要的业务功能。
6. 将增加或者减少费用。

一旦对每项都打了分，就累计每项的得分，得到用例的最后得分。具有最高分的用例指定最高优先级，应该首先开发。

图6-14是会员服务系统的用例分级和评估矩阵的一部分。根据分析的结果，应该首先开发提交订阅读订单用例。但在我们分析完用例依赖关系之前还不能确定这一点。

用例名称	分级标准 (1~5)						总分	优先级	构建周期
	1	2	3	4	5	6			
提交订阅读订单	5	5	5	4	5	5	29	高	1
下新订单	4	4	5	4	5	5	27	高	2
产品查询	1	1	1	1	1	1	6	低	3
建立新会员订阅读程序	4	5	5	3	5	5	27	高	1
每日生成10-30-60天默认合同报告	1	1	1	1	1	1	6	低	3
修改订单	2	2	3	3	4	4	18	中	2

图6-14 用例分级和评估矩阵的一部分

6.4.2 确定用例依赖关系

有些用例依赖于其他用例，其中一个用例使系统处于一种状态，该状态是另一个用例的前置条件。例如，发送俱乐部促销的前置条件是首先创建促销。我们使用**用例依赖关系图**建模这种依赖关系。用例依赖关系图具有以下优点：

- 系统事件及其状态的图形化表述有利于对系统功能的理解。
- 有助于确定遗漏的用例。存在其他用例的执行无法满足的前提条件的用例可能指出遗漏了某个用例。
- 通过描述哪个用例更关键（具有最重要的依赖关系）并需要最高优先权，有助于推动项目管理。

图6-15是图6-14中列出的用例的依赖关系图。依赖于另一个用例的用例使用一条标记“依赖”的虚线连接。在图6-15中，提交订阅读订单用例依赖于建立新会员订阅读程序用例。由于这个依赖关系，应该首先开发建立新会员订阅读程序用例，尽管提交订阅读订单用例在图6-14中具有更高的优先级。

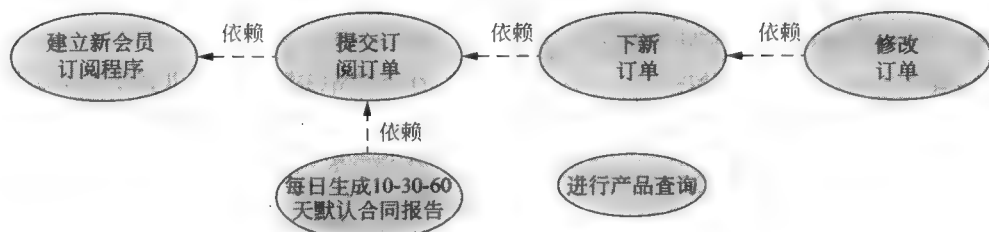


图6-15 用例依赖关系图的例子

① Craig Larman, *Applying UML Patterns* (Upper Saddle River, NJ: Prentice Hall, 1998).

复习题

- 1. 什么是以用户为中心的开发，为什么它对系统开发过程的成功很重要？
- 2. 用例建模如何与以用户为中心的开发关联？
- 3. 除了鼓励用户参与以外，用例建模还提供了许多其他的优点。列出用例建模提供的优点。
- 4. 用例建模使用两个主要的工具——用例图和用例描述。如何使用这两个工具？它们有什么差别？
- 5. 用例图包括三个构件。这三个构件是什么？其用途是什么？
- 6. 在整个系统开发生命周期中如何使用用例？
- 7. 在四类主要角色中，主要的系统角色是谁？
- 8. 在用例图中使用了哪些不同类型的关系？它们各

问题和练习

- 1. 按照作者 Fred Brooks 的说法，在系统开发中要做的最重要的唯一一件事情是什么？用例建模在这个领域有什么帮助？
- 2. 在用例建模中，系统分析员使用哪两种主要工具？描述每种工具并解释它们的用途。
- 3. 在确定和开发用例中，关于用例的用途系统分析员应该总是牢记什么？既然需求调研以前已经完成了，在这时还真的需要花那么多时间在用户身上吗？仅仅一个用例代表着什么？用例同功能需求一样吗？
- 4. 用例首先在系统开发周期的哪个部分被定义？在系统开发周期的什么时候使用，用于什么目的？
- 5. 将下面的关联人员和外部用户同正确的参与者匹配。什么是时序事件？谁和什么被考虑成一个时序事件中的参与者，为什么？

关联人员和外部用户	参与者
美国邮政服务	主要业务参与者
带键盘的计算机化门锁	主要系统参与者
租车代理	外部服务参与者
生成区域销售报告的销售经理	外部接收参与者
自动草地喷灌器	时间
使用 ATM 卡购买汽油的驾驶员	
银行贷款授权服务	

- 6. 以下每个例子属于哪类关系？
 - “打印表格”用例和几个涉及打印不同类型表格的其他用例之间的关系。
 - 骑摩托车的警官和手持式开罚单设备的关系。
 - 客户和销售员的关系，销售员可以查询库存

有什么用途？

- 9. 构造需求用例模型的目的是什么？遵循哪些步骤？
- 10. 为什么确定参与者是用例建模的第一步？
- 11. 当我们寻找业务需求用例时，我们应该知道什么？
- 12. 用例的典型事件过程是什么？
- 13. 为什么分级和评估用例是必需的？
- 14. 用例分级和优先级矩阵中的 6 个评价标准是什么？
- 15. 什么是用例依赖图？为什么使用它？

系统看某项物品是否在库存中，该角色是专门为最小化重复的系统通信创建的。

- “计算 GPA”用例和冗长的“创建副本”用例的关系。
- “传输订单”用例和“下订单”用例的关系。
- 7. Y&J Cookbooks 是一个虚构的企业，它由一对退休的夫妇拥有和运行。到目前为止，Y&J Cookbooks 只通过邮寄订单销售它的图书。所有者现在想开发一个联机系统通过互联网用来销售稀少的和不再发行的食谱。访问者将能够浏览食谱品种，但在购买之前客户需要创建一个客户账号。将只联机接受一种主要信用卡的支付，而且在订单被批准之前要验证信用卡。基于这些信息，确定主要的业务参与者。
- 8. 在用例建模中，一旦你确定了业务参与者，在定义它们中你应该使用什么观点和语言？使用哪种观点和语言构造一份参与者词汇表，以图 6-8 为例。
- 9. 上下文图可以极大地帮助确定不同的用例。为 Y&J Cookbooks 网站准备一份高层上下文图，以图 6-9 为例。
- 10. 在需求用例建模中下一步是确定业务需求用例。每个用例应该捕捉什么？你可以使用何种有效技术确定用例？为了确定用例你将提问什么问题？用例和基本用例之间有什么区别？
- 11. 用例建模的第 3 步是构造用例模型图。根据 Y&J Cookbooks 参与者词汇表和上下文图，创建一个高层用例模型图，展示购物者/客户参与者和系统的交互，包括查询和浏览图书，购买和管理客户账户。确保展示了参与者和每个用例

的关系。

12. 下一步是创建用例描述来记录业务需求。为什么一般在第2步过程中准备用例描述？这两步是什么？基于前面的高层用例模型图，创建一个扩展的描述，以图6-13为例。

项目和研究

1. 在本章开始，引用了一段来自 Frederick P. Brooks Jr. 的文章的文字，他一般被看作是项目管理和软件开发领域的主要作者和共享者之一。在网上搜索这篇文章以及 Fred Brooks 编写的或者与他有关的其他文章。
 - a. 在搜索中，你找到了作者 Fred Brooks 多少篇索引？
 - b. 根据上一章提供的信息，解释 Brooks 的话：“构建一个软件系统的最困难的部分就是正确地确定要构建什么”。
 - c. Brooks 给上述陈述的文章命了什么名字？这篇文章的主题是什么？
 - d. Brooks 最知名的书是什么，该书在第一次出版之后几十年仍在印刷，并被广泛地阅读？这本书的主题是什么？
 - e. 你认为 Brooks 对今天的最大贡献是什么？为什么？
2. 本章提到的 Standish Group 是一个独立的研究小组，他们研究信息技术的变化和趋势。1994年，Standish Group 出版了其奠基性的 CHAOS 报告，这份报告“揭示了如今 MIS 环境下 IT 应用开发项目的全面失败。”从那时起，Standish Group 出版了对原始报告的周期性的修订版本。通过访问它的网站 www.standishgroup.com 中的公共访问区，你可以找到最新的 CHAOS 报告的小结，以及原始的 1994 年版报告。
 - a. Standish Group 使用什么标准来决定一个项目是否成功，质疑，还是失败？
 - b. 基于最新的研究报告，有百分之多少的项目成功，质疑，或是失败？
 - c. 这些最新的比率同本书图 6-1 中所显示的成功相比，质疑和失败比率如何？你将如何描述整个趋势，如果存在的话？
 - d. 基于你的阅读和经验，你相信什么是引起项目成功，质疑，失败比率变化的原因？
 - e. 你认为当前的项目管理和系统开发方法将基本保持不变，但继续优化，还是你预测在未

13. 用例建模和项目管理是什么关系？为什么很重要？为什么用例要分级，使用什么工具给它们分级？谁为分级提供了输入？分级使用什么标准？解释为什么需要确定用例依赖性，举例说明，使用何种工具确定依赖性？

来十几年内会有十分不同的方法出现用于管理项目和开发系统呢？

3. 选择一个在你的企业或学校中使用的信息系统。同熟悉这个系统的系统分析员或设计人员交谈。根据提供的信息，完成以下工作：
 - a. 描述你选择的信息系统和企业。
 - b. 创建系统的上下文图。
 - c. 确定业务参与者。
 - d. 创建一份参与者词汇表。
 - e. 确定业务需求基本用例。
 - f. 创建一份用例词汇表。
4. 基于上一题中提供的涉及你选择的系统的信息：
 - a. 构造一个包括所有主要子系统的用例模型图。
 - b. 描述基本用例中的三个扩展用例。
 - c. 准备一份用例分级和优先级矩阵，然后使用它来分级和评估用例。
 - d. 确定用例依赖关系。
 - e. 准备一份用例依赖图。
5. 搜索网络或者你学校图书馆的专业杂志，查找关于用例建模新的和正出现的发展的文章。选择两篇文章，然后：
 - a. 为每篇文章提供一份参考书目（使用你学校使用的格式）。
 - b. 用你自己的话为每篇文章创建一份摘要。
 - c. 比较每篇文章中描述的方法。描述你觉得哪一个更可行和/或更重要？解释为什么？
6. 同几个开发人员交谈，讨论他们对用例建模的看法。如果可能，试着找到来自不同企业和/或具有不同长度工作经历，以及不同类型经历的开发人员（即一个具有作为系统分析员经历的开发人员）。
 - a. 依据他们的经验描述与你交谈的开发人员。例如：他们在 IT 行业工作了多长时间，他们的专业领域是什么，他们对用例建模有多熟悉？
 - b. 他们的企业特征是什么？

- c. 你问了哪些问题?
- d. 每个开发人员对于用例建模的重要性和价值各持什么观点?
- e. 这些开发人员在他们目前的工作中真的使用了用例建模吗?为什么?
- f. 如果让他们当其企业的 CIO 一天,他们会改变该企业关于用例建模的 IT 架构吗?
- g. 使用能力成熟度模型,你将如何评估那些企业的成熟度等级?为什么?
- h. 你能从涉及用例建模的实际应用的交谈中得出什么结论?
- i. 在交谈之前你对用例建模的重要性和价值持什么看法?交谈后你的看法有所改变吗?如果有,是如何改变的?为什么?

小型案例

1. 在第 5 章的一个小型案例中,你就一个联机课程注册系统与关联人员交谈。在那个练习中,你要理解那些关联人员对这个系统的问题和需求。回顾你从那些交谈中得到的发现。
 - a. 访问其他学校的注册系统。存在功能或者易用性的差别吗?存在你认为关联人员实际上会喜欢/不喜欢的特征吗?记录并截屏其他系统的例子。
 - b. 同你原先交谈过的那些关联人员进行一次会谈,确定对你的学校的功能需求和使用需求。
2. 为在前面问题中找到的至少一个功能需求,创建一个用例描述。以图 6-10 为例。
3. 确定学校注册系统所有的参与者。每个参与者将启动哪些用例?
4. 使用你对上一问题的回答,绘制一张学校注册系统的用例图。

团队和个人练习

1. 圆桌讨论:你认为人们在回答面谈问题时总是绝对诚实的吗?
2. 观看一个无声电影。圆桌讨论:在言语之外发生了什么交流?
3. 圆桌讨论:当你通过类似面谈的方法确定一个系统的需求时,假设正在同你面谈和收集信息的人希望系统成功。情况总是这样的吗?如果不是这样,你将如何处理需求收集?

数据建模和分析

本章概述和学习目标

在本章中，你将学到如何使用流行的数据建模工具（实体关系图）记录系统必须收集和存储的数据，而独立于如何显示或如何使用数据——也就是说，独立于特定的输入、输出和处理。你还将学到一种称为规范化（normalization）的数据分析技术，这个技术用于确保数据模型成为一个“好的”数据模型。

在本章中，你将学到以下内容：

- 定义系统建模并区分逻辑和物理系统模型。
- 定义数据建模并解释其优点。
- 认识并理解数据模型的基本概念和结构。
- 阅读并解释实体关系数据模型。
- 解释在项目开发期间的什么时候构造数据模型以及在哪里存储数据模型。
- 获取实体和关系。
- 构造实体关系上下文图。
- 确定实体的键，并构造基于键的模型图。
- 构造包含全部属性的实体关系图，对所有的数据结构和属性加以描述并纳入到资料库或百科全书中。
- 规范化逻辑数据模型，消除可能使数据库不稳定、不灵活和不可扩展的不纯洁性。
- 描述一种将数据需求映射到业务操作地点的有用工具。

本章关键术语

实体关系图（Entity Relationship Diagram, ERD）是一种利用符号记法按照数据描述的实体和关系来刻画数据的数据模型。

实体（entity）是我们需要收集数据和存储数据的人、地点、对象、事件或概念的类。

实体实例（entity instance）是实体的具体值。

属性（attribute）是实体的描述性性质或特征。同义词包括要素、性质和域。

组合属性（compound attribute）实际上是由其他属性构成的属性。它在不同的数据建模语言中有很多同义词：串联属性、合成属性和数据结构。

数据类型（data type）是属性的一个参数，定义了这个属性中可以存储什么类型的数据。

域（domain）是属性的一个参数，定义了这个属性可以取的合法值。

默认值（default value）是如果用户没有指定值的话将被记录的值。

键（key）是一个属性（或一组属性），它们对每个实体实例具有一个唯一的值。有时也称为标识符。

复合键（concatenated key）是唯一地标识实体的一个实例的一组属性。同义词包括组合键和合成键。

候选键（candidate key）是一组可以作为一个实体的主键的键。有时称为候选标识符。

主键（primary key）是最常用来唯一地确定一个实体实例的候选键。

替代键（alternate key）是没有被选中作为主键的任何候选键。

子集准则（subsetting criteria）是一个属性（或合成属性），其有限的取值范围把所有的实体实例分成了有用的子集。这有时也称为反向条目。

关系（relationship）是存在于一个或多个实体之间的自然业务联系。

基数 (cardinality) 定义了一个实体相对于另一个关联实体的某个具体值的最小和最大具体值数量。

度数 (degree) 是参与那个关系的实体数量。

递归关系 (recursive relationship) 是存在于同一个实体的不同实例之间的关系。

关联实体 (associative entity) 是一个从多个其他实体继承其主键的实体。

外键 (foreign key) 是一个实体的主键，它被贡献给（复制到）另一个实体以确定一个关系实例。

非确定性关系 (nonidentifying relationship) 是每个参与关系的实体都有各自的独立主键的关系。

确定性关系 (identifying relationship) 是父实体贡献其主键成为子实体的主键的一部分的关系。

非特定关系 (nonspecific relationship) 是一个实体的多个实例同另一个实体的多个实例相关联的关系，也称为多对多关系。

泛化 (generalization) 是指将几类实体公共的属性组合成独立的实体。

超类 (supertype) 是一个实体，其实例存储了一个或多个实体子类的公共属性，也称抽象类或父类。

子类 (subtype) 是一个实体，其实例从一个实体超类中继承了一些公共属性。

应用数据模型 (application data model) 单个信息系统的数据模型。

上下文数据模型 (context data model) 是包括实体和关系，但不包括属性的数据模型。

基于键的数据模型 (key-based data model) 是包括带有精确基数的实体和关系的数据模型，它将非特定关系分解成关联实体，还包括进主键和替代键。

具有完整属性的数据模型 (fully attributed data model) 是包含了所有实体、属性、关系、子集准则和精确基数的数据模型。

智能键 (intelligent key) 是一个业务编码，其结构表达了一个实体实例的数据。

数据分析 (data analysis) 是为将数据模型实现成数据库而改进数据模型的技术。

规范化 (normalization) 是一种数据分析技术，该技术组织数据属性以便它们可以组合起来形成无冗余的、稳定的、灵活的并具有适应性的实体。

第一范式 (First Normal Form, 1NF) 实体的所有属性对于实体的单个实例都只具有一个值。

第二范式 (Second Normal Form, 2NF) 实体的所有非主键属性的值都依赖于主键。

第三范式 (Third Normal Form, 3NF) 实体的非主键属性的值不依赖于任何其他非主键属性。

导出属性 (derived attribute) 是其值可以从其他属性中计算出来或者可以从其他属性值通过逻辑导出的属性。

传递依赖关系 (transitive dependency) 是当一个非键属性的值依赖于另一个非键属性时（而不是导出）就存在的一种关系。

数据-地点-CRUD 矩阵 (data-to-location-CRUD matrix) 是用来将数据需求映射到地点的矩阵。

7.1 数据建模简介

系统模型在系统开发中扮演着一个重要的角色。本章将介绍**数据建模**，它是一种为数据库定义业务需求的技术。因为数据模型最终要实现成数据库，所以数据建模有时称为**数据库建模**。

图 7-1 中的例子是一个简单的数据模型，称为**实体关系图 (ERD)**。这幅图做出了以下的业务断言：

- 我们需要存储关于 CUSTOMER（客户）、ORDER（订单）和 INVENTORY PRODUCT（库存产品）的数据。
- CUSTOMER NUMBER 的值唯一地确定了且仅仅确定了一个 CUSTOMER。ORDER NUMBER 的值唯一地确定了且仅仅确定了一个 ORDER。PRODUCT NUMBER 的值唯一地确定了且仅仅确定了一个 INVENTORY PRODUCT。
- 对于客户，我们需要知道 CUSTOMER NAME、SHIPPING ADDRESS、BILLING ADDRESS 和 BALANCE DUE。对于订单，我们需要知道 ORDER DATE 和 ORDER TOTAL COST。对于库存产品，我们需要知道 PRODUCT NAME、PRODUCT UNIT OF MEASURE 和 PRODUCT UNIT PRICE。
- 客户已经发出了零个、一个或者多个订单。

- 订单完全由客户发出。CUSTOMER NUMBER 的值（记录在订单中）确定了那个客户。
- 一个订单出售一个或多个 ORDERED PRODUCT（订购的产品）。这样，一个订单应该至少包含一个订购的产品。
- 库存产品可能已经作为零个、一个或者多个订购的产品被售出。
- 订购的产品确定了订单上的一个库存产品。ORDER NUMBER（对订购的产品）确定了这个订单，并且 PRODUCT NUMBER（对订购的产品）确定了库存产品。合在一起，它们确定且仅确定了一个订购的产品。
- 对于每个订购的产品，我们需要知道 QUANTITY ORDERED 和 UNIT PRICE AT TIME OF ORDER。

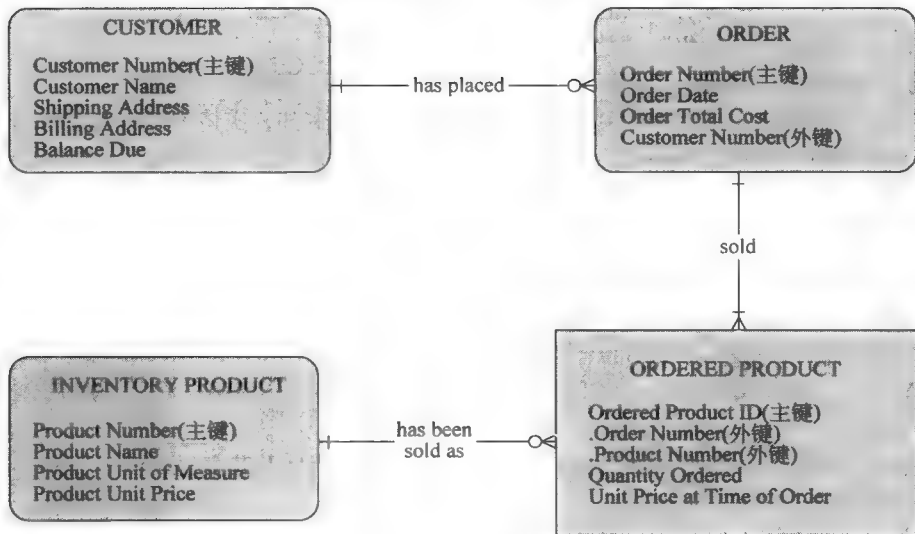


图 7-1 一个实体关系数据模型

在学完本章后，你将能够阅读并构造数据模型。

7.2 数据建模的系统概念

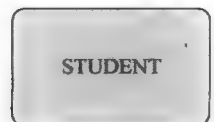
用于数据建模的符号记法有多种类型。实际的模型经常称为**实体关系图**（Entity Relationship Diagram, ERD），因为它按照数据描述的实体和关系来刻画数据。ERD 有好多种符号记法，大多数记法都以它们的发明者命名（例如，Chen、Martin、Bachman、Merise），或者采用公布的标准命名（例如，IDEF1X）。这些数据建模“语言”一般都支持相同的基本概念和结构。我们采用 Martin（信息工程）符号记法，因为它应用广泛，并且得到 CASE 工具的支持。

首先介绍一些基本概念，这些基本概念是所有数据模型的基础。

7.2.1 实体

所有系统都包括数据，通常包括大量的数据。数据描述了“事物”。考虑一个学校系统。学校系统包括了描述各种事物的数据，例如“学生”、“教师”、“课程”和“教室”。对于这些事物，不难想像出一些描述其任意给定实例的数据。例如，描述一个特定学生的数据可能包括“姓名”、“地址”、“电话号码”、“出生日期”、“性别”、“民族”、“专业”和“平均积分点”等等。

我们需要一个概念来抽象地表示一组类似事物的所有实例，我们称这个概念为**实体**。实体是指某些事物，企业需要存储有关这些事物的数据。在系统建模中，我们发现为每个抽象概念指定一个表示形状是有用的。在本书中，一个实体将绘制成一个圆角矩形（见右图），这个形状表示了命名实体的所有实例。例如，实体 STUDENT（学生）表示系统中的所有学生。因此，一个实体确定了特定的



一个实体

一类事物，有别于其他实体。

实体的分类（和例子）包括：

- 人：“代理”、“承包人”、“客户”、“部门”、“分部”、“雇员”、“导师”、“学生”、“供应商”。注意，人实体类可以表示个人、小组或组织。
- 地点：“销售地区”、“建筑物”、“房间”、“分支办公室”、“校园”。
- 对象：“图书”、“机器”、“部件”、“产品”、“原材料”、“软件许可证”、“软件包”、“工具”、“汽车模型”、“汽车”。对象实体可以表示实际的对象（例如，软件许可证）或者一类对象的说明（例如，不同的软件包的说明）。
- 事件：“应用”、“奖励”、“取消”、“分类”、“飞行”、“开发票”、“订单”、“注册”、“续借”、“获取”、“预订”、“销售”、“旅行”。
- 概念：“账号”、“时间段”、“债券”、“课程”、“基金”、“资格”、“股票”。

区分实体及其实例很重要。**实体实例**是指实体的具体值。例如，实体 STUDENT 可以拥有多个实例：Mary、Joe、Mark、Susan、Cheryl 等。在数据建模中，我们不关心单个学生，因为我们认为每个学生都用类似的数据描述。

7.2.2 属性

如果一个实体是我们想存储数据的某样东西，那么就需要确定我们想存储关于一个给定实体的每个实例的什么数据。我们称这些数据为**属性**。正如本节开始注意到的，实体 STUDENT 的每个实例都可以用以下属性描述：NAME、ADDRESS、PHONE NUMBER、DATE OF BIRTH、GENDER、RACE、MAJOR、GRADE POINT AVERAGE 等。

我们现在可以扩展实体图形抽象，在实体形状内记录这些属性及其名字，从而把属性引入进来（见右图）。

某些属性可以被逻辑上组合成超级属性，称为**组合属性**。例如，学生的 NAME 属性实际上是一个组合属性，它由 LAST NAME、FIRST NAME 和 MIDDLE INITIAL 构成。在右图中，我们演示了组合属性的一个可能的符号记法。注意，被包含在组合属性中的每个原始属性前面有一个小点。

7.2.2.1 域

当分析系统时，我们应该为属性定义合法的或者有业务含义的值。每个属性的值按照三种性质定义：数据类型、域和默认值。

属性的**数据类型**定义了这个属性中可以存储什么类型的数据。为了便于系统分析和业务需求定义，为业务属性声明逻辑（非技术性的）数据类型是有用的。为了表述方便起见，我们将使用表 7-1 所示的逻辑数据类型。



属性和组合属性

表 7-1 属性的有代表性的逻辑数据类型

逻辑数据类型	逻辑业务含义
NUMBER	任何数、实数或整数
TEXT	一个字符串，包括数字。当数字包含在 TEXT 属性中时，意味着我们不希望进行那些数字的算术或比较运算
MEMO	同 TEXT 一样，但具有不确定的大小。某些业务系统要求能够附加潜在的长注解信息到一个给定的数据库记录中
DATE	任何格式的日期
TIME	任何格式的时间
YES/NO	只能取这两个值中的一个值的属性
VALUE SET	一个有限值集合。在大多数情况下，应该建立一个编码方案（例如，FR = freshman、SO = sophomore、JR = junior、SR = senior 等）
IMAGE	任何图形或图像

属性的数据类型约束了它的域。属性的域定义了这个属性可以取的合法值。最终，系统设计人员必须使用相应技术来强制所有属性的业务域。表 7-2 演示了如何表达每个数据类型的逻辑域。

表 7-2 逻辑数据类型的有代表性的逻辑域

数据类型	域	例 子
NUMBER	对于整数，指定范围： {最小~最大} 对于实数，指定范围和精度： {精度最小值~精度最大值}	{10~99} {1.000~799.999}
TEXT	TEXT（属性的最大长度） 实际值通常是无限的，但是用户可以指定某个较小的限制范围	TEXT（30）
MEMO	不可应用。对长度或内容没有逻辑限制	不可应用
DATE	MMDDYYYY 格式的变量。为了兼容 2000 年，不要把年简写成 YY。 一般不存储格式化字符；所以，不要包括冒号或者连字符	MMDDYYYY MMYYYY YYYY
TIME	对于 AM/PM 时间：HHMMT 或 对于军队时间：HHMM	HHMMT HHMM
YES/NO	{YES, NO}	{YES, NO} {ON, OFF}
VALUE SET	{值 1, 值 2, ..., 值 n} 或 {表示代码及含义的表}	{ FRESHMAN, SOPHOMORE, JUNIOR, SENIOR} { FR = FRESHMAN SO = SOPHOMORE JR = JUNIOR SR = SENIOR}
IMAGE	不可应用，但是图像的特征最终将对设计人员有用	不可应用

最后，每个属性应该有逻辑默认值。属性的默认值如果是用户没有指定值的话将被记录在属性中的值。表 7-3 显示了一个属性可能的默认值。注意，NOT NULL 说明了属性的每个实例都必须有值，而 NULL 说明了属性的某些实例可以有值或者没有值。

表 7-3 属性允许的默认值

默 认 值	解 释	例 子
来自域中的一个合法值（如上所述）	对于这个属性的一个实例，如果用户没有指定值，那么就用这个值	0 1.00 FR
NONE 或 NULL	对于这个属性的一个实例，如果用户没有指定值，那么就保持为空	NONE NULL
REQUIRED 或者 NOT NULL	对于这个属性的一个实例，要求用户输入一个来自域中的合法值（适用于没有足够通用的值可以用作默认值而又必须输入一些值的情况）	REQUIRED NOT NULL

7.2.2.2 标识符

一个实体包括很多实例，也许成千上万个。有必要根据一个或多个属性的数据值唯一地标识每个实例。因此，每个实体必须具有一个标识符或键。例如，实体 STUDENT 的每个实例可以通过键 STUDENT NUMBER（学号）属性唯一地标识。没有两个学生会具有相同的 STUDENT NUMBER。

有时则需要多个属性来唯一地标识一个实体实例。由一组属性构成的键称为复合键。例如，在视频商店中每个 DVD 实体实例可以通过组合 TITLE NUMBER 和 COPY NUMBER 属性唯一地标识。仅仅有 TITLE NUMBER 是不够的，因为商店可能拥有一个视频产品号的多个拷贝；仅仅有 COPY NUMBER 也

是不够的，因为我们事先假定拥有的每个视频产品号都有一个 COPY #1。我们需要同时使用这两个数据来确定一个特定的磁带（例如，COPY#7 和 Star Wars: Revenge of the Sith）。在本书中，我们将为属性组及其单个属性命名。例如，DVD 的复合键将记录成如下形式：

DVD ID

. TITLE NUMBER

. COPY NUMBER

一个实体可能有多个键。例如，实体 EMPLOYEE 可以由 SOCIAL SECURITY NUMBER 唯一地确定，或者由公司指定的 EMPLOYEE NUMBER 或 E-MAIL ADDRESS 唯一地确定。这些属性都被称为候选键。候选键是一个实体实例的“主键的候选键”（候选键可以是一个属性或者一个复合键）。主键是最常用来唯一地确定一个实体实例的候选键。主键的默认值总是 NOT NULL，如果键没有值，它就不能起到标识实体实例的作用了。没有被选中作为主键的任何候选键都称为替代键。常用的同义词是次键。在图 7-4 中，我们演示了主键和替代键的符号记法。所有候选键不是主键就是替代键，所以我们没有为候选键再规定一个专门符号。不属于主键一部分的所有属性被称为非键值属性。

有时，也需要标识出实体实例的一个子集，而非单个实例。例如，我们可能需要使用一种简单的方式来确定所有男生和所有女生。子集准则是一个属性（或合成属性），其有限的取值范围把所有的实体实例分成了有用的子集。这有时也称为反向条目。在 STUDENT 实体中，属性 GENDER 把 STUDENT 的实例分成了两个子集：男生和女生。通常，仅当一个属性具有有限（意思是可数的）数量的合法值时，子集准则才有效。例如，GRADE POINT AVERAGE 就不是一个有效的子集准则，因为那个属性在 0.00 ~ 4.00 之间有 999 个可能的值。右图也演示了子集准则的符号。

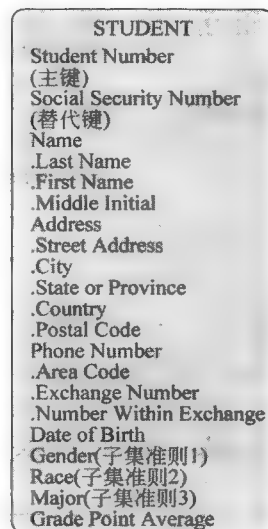
7.2.3 关系

从概念上说，实体和属性都不是孤立存在的。它们各自代表的事物互相交互，并且互相影响，共同支持业务任务。下面，我们介绍关系的概念。关系是存在于一个或多个实体之间的自然业务联系。关系可以表示链接实体的一个事件，或者仅仅是存在于实体之间的逻辑关系。例如，考虑实体 STUDENT 和 CURRICULUM。我们可以做出链接学生和课程的业务推断：

- 一个学生注册一门或多门课程。
- 一门课程正被零个、一个或者多个学生学习。

动词短语定义了两个实体之间的业务关系。

我们可以用图形表示 STUDENT 和 CURRICULUM 之间的关系，如图 7-2 所示。连接线表示了一个关系，动词短语描述了这个关系。注意所有的关系隐含地都是双向的，意味着它们可以从两个方向上解释（如上面的业务推断所暗示的那样）。数据建模方法可能在关系的命名上会有所不同——有些包括两个动词，而另一些仅包括一个动词。



键和子集准则

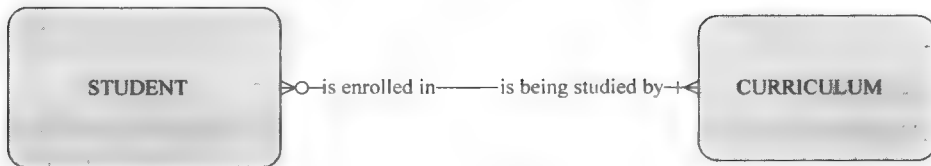


图 7-2 关系（多对多）

7.2.3.1 基数

图 7-2 还显示了每个关系的复杂度或度数。例如，在上面的业务推断中，我们还必须回答以下问题：

- 对应课程的每个实例都必然存在学生的一个实例吗？不！
- 对应学生的每个实例都必然存在课程的一个实例吗？是的！
- 对应学生的每个实例存在多少个课程实例？许多！
- 对应课程的每个实例存在多少个学生实例？许多！

我们称这个概念为**基数**。**基数**定义了一个实体相对于另一个关联实体的某个具体值的最小和最大具体值数量。因为所有的关系都是双向的，所以对于每个关系在两个方向上都必须定义基数。图 7-3 表示了一种流行的基数图形化符号，基数符号的例子也在图 7-2 中进行了演示。



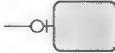



基数含义	最小实例数	最大实例数	图形化符号
正好一个（一个且仅一个）	1	1	 或 
零个或一个	0	1	
一个或多个	1	多个 (>1)	
零个、一个或多个	0	多个 (>1)	
大于一个	>1	>1	

图 7-3 基数的符号

从概念上说，基数告诉我们在图 7-2 中有关想存储的数据的以下规则：

- 当我们在数据库中插入一个 STUDENT 实例时，我们必须链接（关联）那个 STUDENT 实例到 CURRICULUM 的至少一个实例。按业务词汇表述就是：“不申报一个专业就不能接纳一个学生。”（大多数学校将包括 CURRICULUM 的一个称为“未决定的”或“未申报的”的实例。）
- 一个学生可以学习多门课程，而且我们必须能够存储指示一个给定学生的所有课程的数据。
- 在我们可以链接（关联）学生到一门课程之前，我们必须在数据库中插入那门课程。这就是为什么一门课程可以有零个学生——还没有学生被允许上哪门课程。
- 一旦一门课程已经插入到数据库中，我们就可以链接（关联）许多学生到那门课程。

7.2.3.2 度数

数据关系复杂性的另一个度量是它的度数。关系的**度数**是参与那个关系的实体数量。到目前为止，我们讨论的所有关系都是二维的（度数=2）。换句话说，有两个不同的实体参与这个关系。

关系也可以存在于同一个实体的不同实例之间，我们称这种关系为**递归关系**（度数=1）。例如，在你所在的学校中，一门课可能是其他课程的前置条件；类似地，一门课也可能有几门其他课程作为它的前置条件。图 7-4 演示了这种多对多的递归关系。

关系还可以存在于两个以上的不同实体之间，这种关系有时被称为 N 维关系。图 7-5 演示了一个三维关系。N 维关系用一个新的称为**关联实体**的实体结构说明。关联实体是一个从

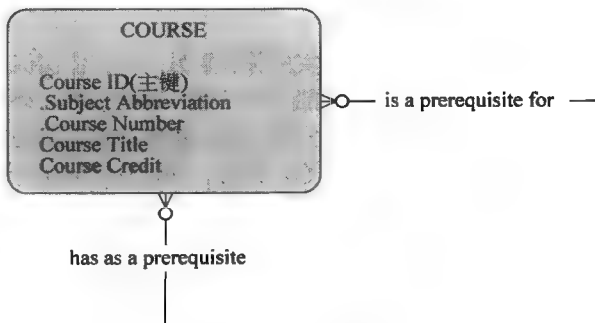


图 7-4 递归关系

多个其他实体（称为父实体）继承其主键的实体，其复合键的每个部分指向每个连接实体的一个且仅一个实例。

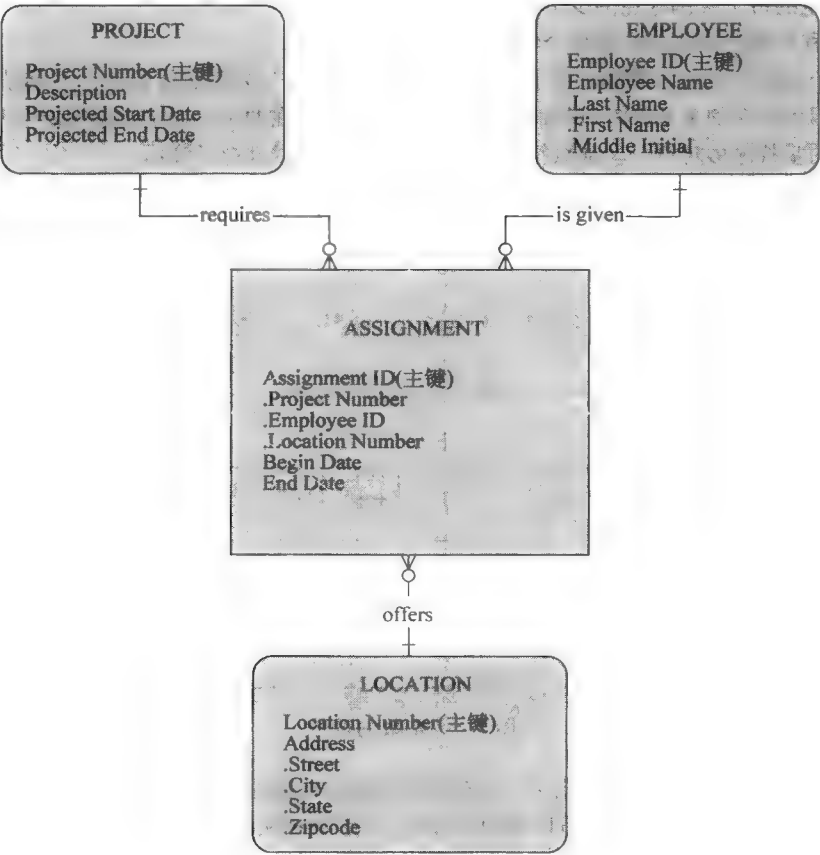


图 7-5 三维关系

在图 7-5 中，关联实体 ASSIGNMENT（注意其独特的形状）与一个 EMPLOYEE 实体、一个 LOCATION 实体和一个 PROJECT 实体匹配。对于 ASSIGNMENT 实体的每个实例，键指出了哪个 EMPLOYEE ID、哪个 LOCATION NUMBER 和哪个 PROJECT NUMBER 被组合形成这一任务。

同样，如图 7-5 所示，一个关联实体也可以通过其自身的非键属性描述。除了主键以外，ASSIGNMENT 实体由属性 BEGIN DATE 和 END DATE 描述。如果你仔细考虑它，这些属性没有一个描述了 EMPLOYEE 实体、LOCATION 实体或 PROJECT 实体的内容——它们描述了那些实体实例之间的一个关系实例。

7.2.3.3 外键

关系隐含了一个实体实例关联于另一个实体实例。我们应该能够为任何给定的实体确定这些实例。确定特定相关实体实例的能力涉及到外键的创建。外键是一个实体的主键，它被贡献给（复制到）另一个实体以确定一个关系实例。子实体中的外键总是匹配父实体中的主键。在图 7-6a 中，我们用简单的数据模型演示了外键概念。注意，DEPARTMENT 实体的最大基数是“1”，而 CURRICULUM 实体的最大基数是“许多”。在这种情况下，DEPARTMENT 实体称为父实体，CURRICULUM 实体称为子实体。主键总是由父实体贡献给子实体作为外键。这样，CURRICULUM 实体的一个实例现在有了一个外键 DEPARTMENT NAME，其值指向提供那个课程的 DEPARTMENT 实例（外键从来不会从子实体贡献给父实体）。

在我们的例子中，CURRICULUM 实体和 DEPARTMENT 实体之间的关系是一个非确定性关系。非

确定性关系是每个参与关系的实体都有各自的独立主键的关系。换句话说，不共享主键属性。实体 CURRICULUM 和 DEPARTMENT 也被称为强实体或者独立实体，因为任一个实体都不依靠另一个实体作为自己的标识。但有时，一个外键可以参与子实体作为其主键的一部分。例如，在图 7-6b 中，父实体 BUILDING 贡献其键给实体 ROOM。因此，BUILDING NAME 就作为关联 ROOM 实体和 BUILDING 实体的一个外键，并同 ROOM ID 属性一起唯一地标识 ROOM 实体的一个实例。在这些情况下，父实体和子实体之间的关系被称为确定性关系。确定性关系是父实体贡献其主键成为子实体的主键的一部分的关系。任何确定性关系中的子实体经常被称为弱实体，因为其存在依赖于父实体的存在。

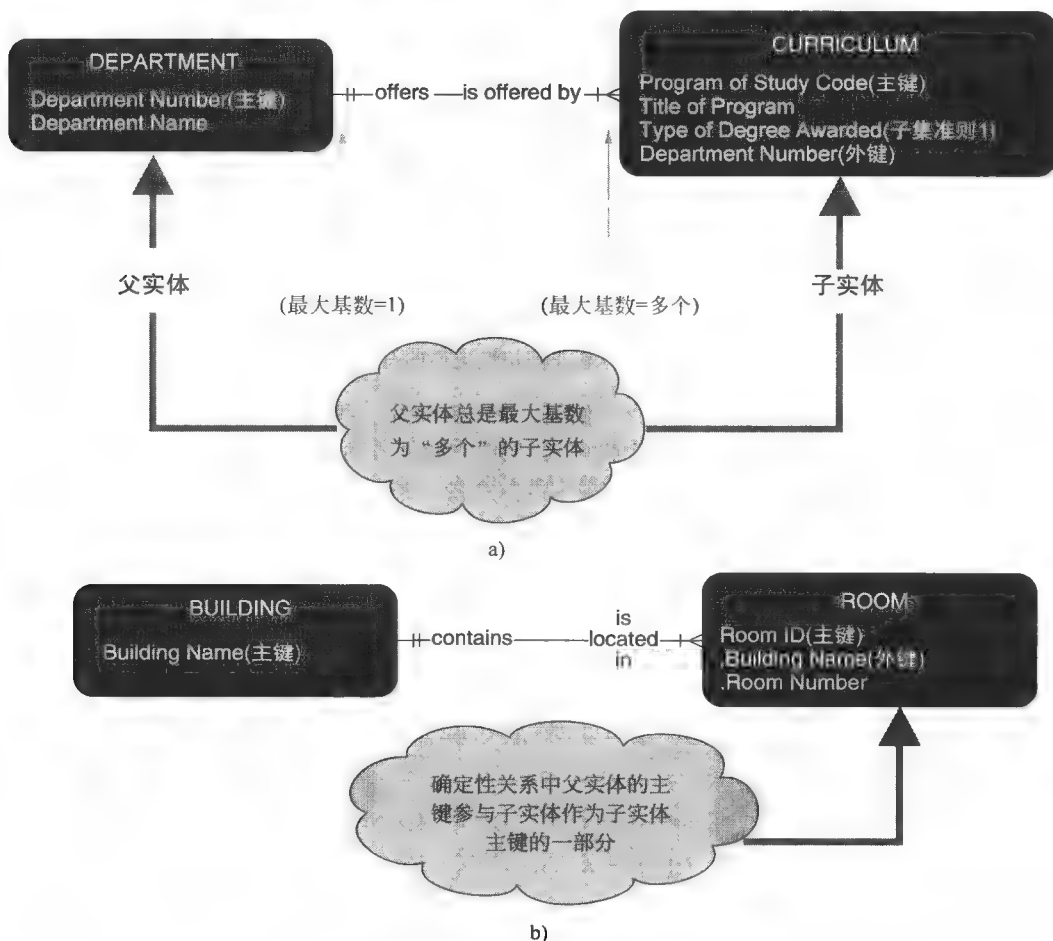


图 7-6 外键

大多数流行的 CASE 工具和数据建模方法都使用不同符号记法来区分确定性关系和非确定性关系，以及强实体和弱实体。在图 7-7 中，我们使用虚线符号表示 PASSENGER 实体和 SEAT ASSIGNMENT 实体之间的非确定性关系。因为 SEAT ASSIGNMENT 实体的部分主键是来自父实体 FLIGHT 的外键 FLIGHT NUMBER，所以这个关系就是一个确定性关系，并用实线表示。最后，SEAT ASSIGNMENT 是一个弱实体，因为它接受 FLIGHT 的主键以构成其主键。弱实体使用在一个圆角矩形内的圆角矩形组合符号表示。

注意 为了进一步强化以上的确定性关系和非确定性关系，以及强实体和弱实体的概念，也为了与大多数流行的数据建模方法及大多数广泛使用的 CASE 工具相一致，作者在本书提供的所有后续数据建模例子中都使用以上的建模符号记法。

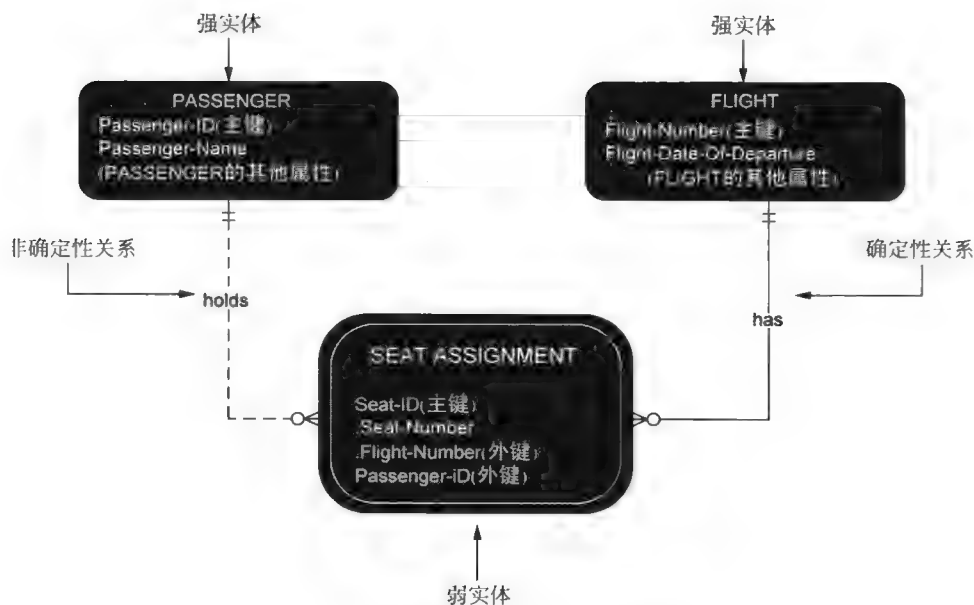


图 7-7 弱实体和非确定性关系的符号表示

如果你无法区分父实体和子实体怎么办？例如，在图 7-8a 中，我们看到一个 CURRICULUM 实体实例注册了零个、一个或者多个 STUDENT 实体实例。同时，我们看到一个 STUDENT 实体实例被注册在一个或多个 CURRICULA 实体实例中。每边的最大基数都是“多个”。那么，哪个是父实体？哪个是子实体？你无法确定！这称为非特定关系。非特定关系（或多对多关系）是一个实体的多个实例同另一个实体的多个实例相关联的关系。这种关系仅仅适用于初始数据模型中，并应该尽快地分解掉。

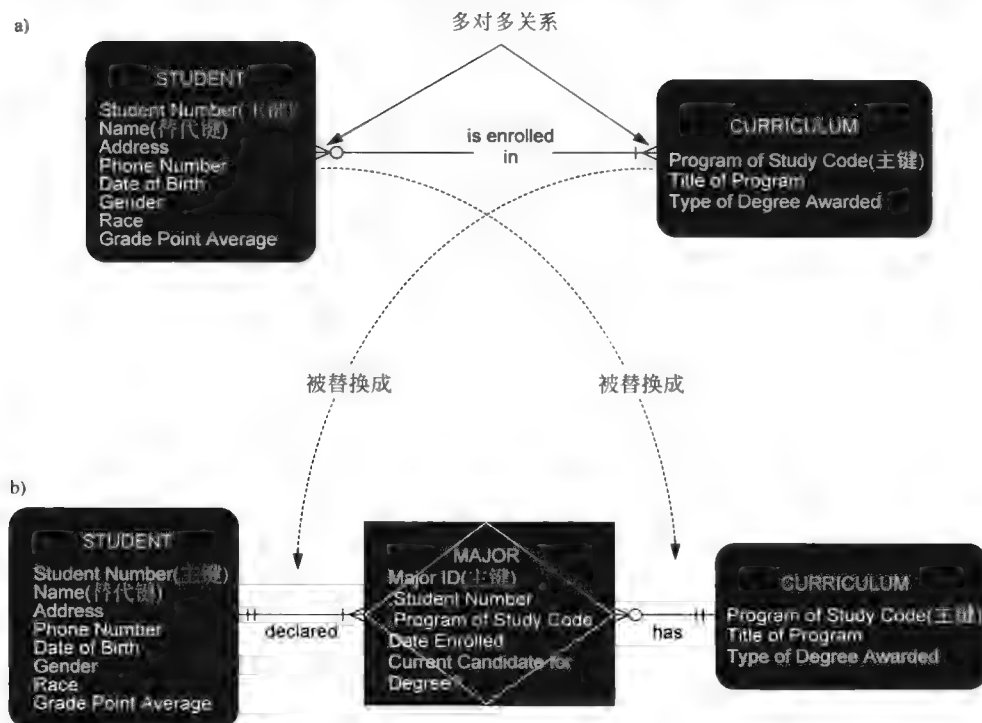


图 7-8 用一个关联实体分解非特定关系

许多非特定关系可以分解成两个一对多关系。如图 7-8b 所示，每个实体都成为一个父实体，一个新的关联实体被引入作为每个父实体的子实体。在图 7-8b 中，MAJOR 实体的每个实例表示了一个学生注册一门课程。如果一个学生申请了两个专业 (major)，那么该学生就将拥有两个 MAJOR 实体实例。

仔细地研究图 7-8b。对于关联实体，从子实体到父实体的基数总是一个且仅一个。这意味着 MAJOR 实体的一个实例必须对应一个且仅一个学生和一个且仅一门课程。从父实体到子实体的基数取决于业务规则。在我们的例子中，一个学生必须申报一个或多个专业。相反，一门课程正在零个、一个或多个专业中学习，也许它是一门新的课程，还没有人接受它。关联实体也可以使用它自己的非键属性描述（例如，DATE ENROLLED 和 CURRENT CANDIDATE FOR DEGREE 属性）。最后，关联实体继承了父实体的主键，因此所有关联实体都是弱实体。

并非所有的非特定关系都可以并且应该以上面所述的方式被自动地分解。偶尔，非特定关系是由于分析员错误地确定其他实体的存在而造成的。例如，检查图 7-9a 中实体 CUSTOMER 和 PRODUCT 之间的关系 “orders” 建议了一个用户可能想存储数据的事件！这个事件代表了一个称为 ORDER 的事件实体，如图 7-9b 中所示。在现实中，实体

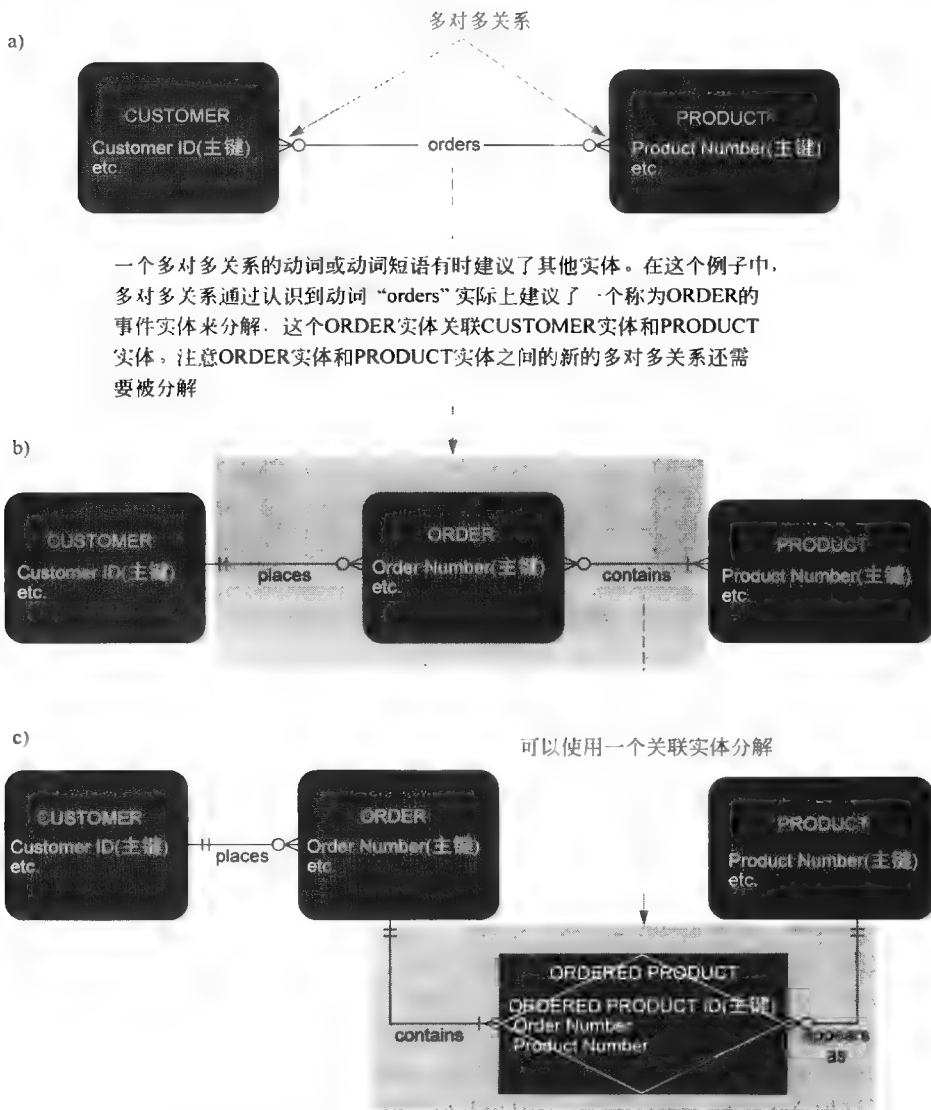


图 7-9 通过识别一个基本业务实体来分解非特定关系

CUSTOMER 和 PRODUCT 之间不存在图 7-9a 中所描绘的自然而直接的关系。相反，它们是通过一个 ORDER 实体的形式间接相关的。因此，我们的多对多关系被 CUSTOMER 实体、ORDER 实体和 PRODUCT 实体之间的独立关系所代替。注意，实体 ORDER 和 PRODUCT 之间的关系是多对多关系。这个关系将需要通过把它替换成图 7-9c 中展示的一个关联实体和两个一对多关系来分解。

最后，某些非特定关系可以通过引入独立关系来分解。注意图 7-10a 中的 TRANSFER（转账）实体和 BANK ACCOUNT（银行账号）实体之间的多对多关系。虽然一个转账事务包含许多银行账号，而且一个银行账号可以包含到多个转账事务中，但我们必须小心！数据建模符号有时可能会误导我们。从技术角度来说，一个转账事务包含两个银行账号。当我们知道一个多对多关系特定的最大出现次数时，就常常表明我们原来的关系很弱或者太一般了。注意在图 7-10b 中，关系“包含”（involves）由两个独立的一对多关系代替，它们更精确地描述了实体 TRANSFER 和 BANK ACCOUNT 之间的业务关系。

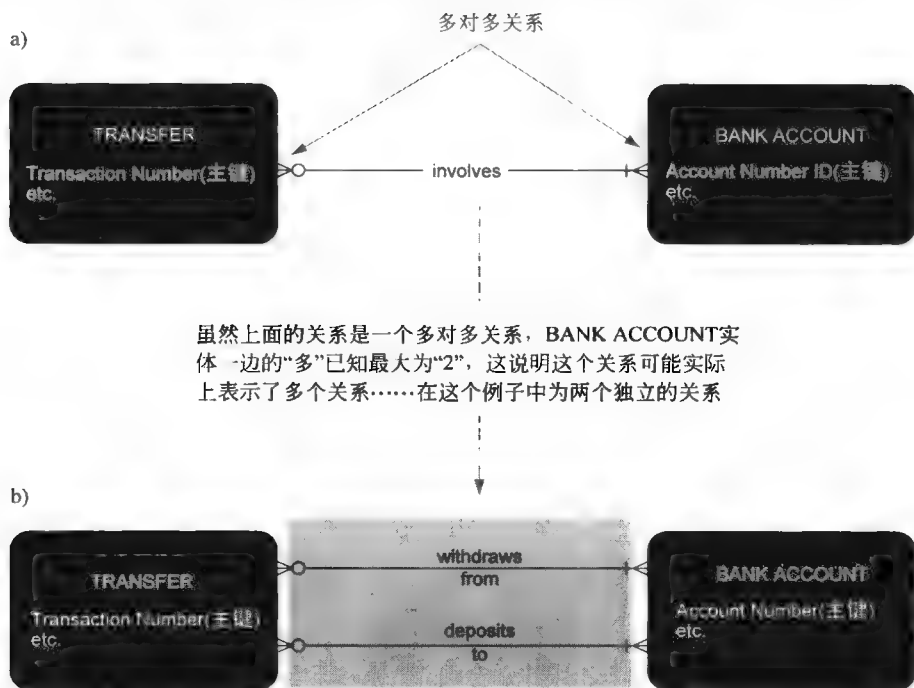
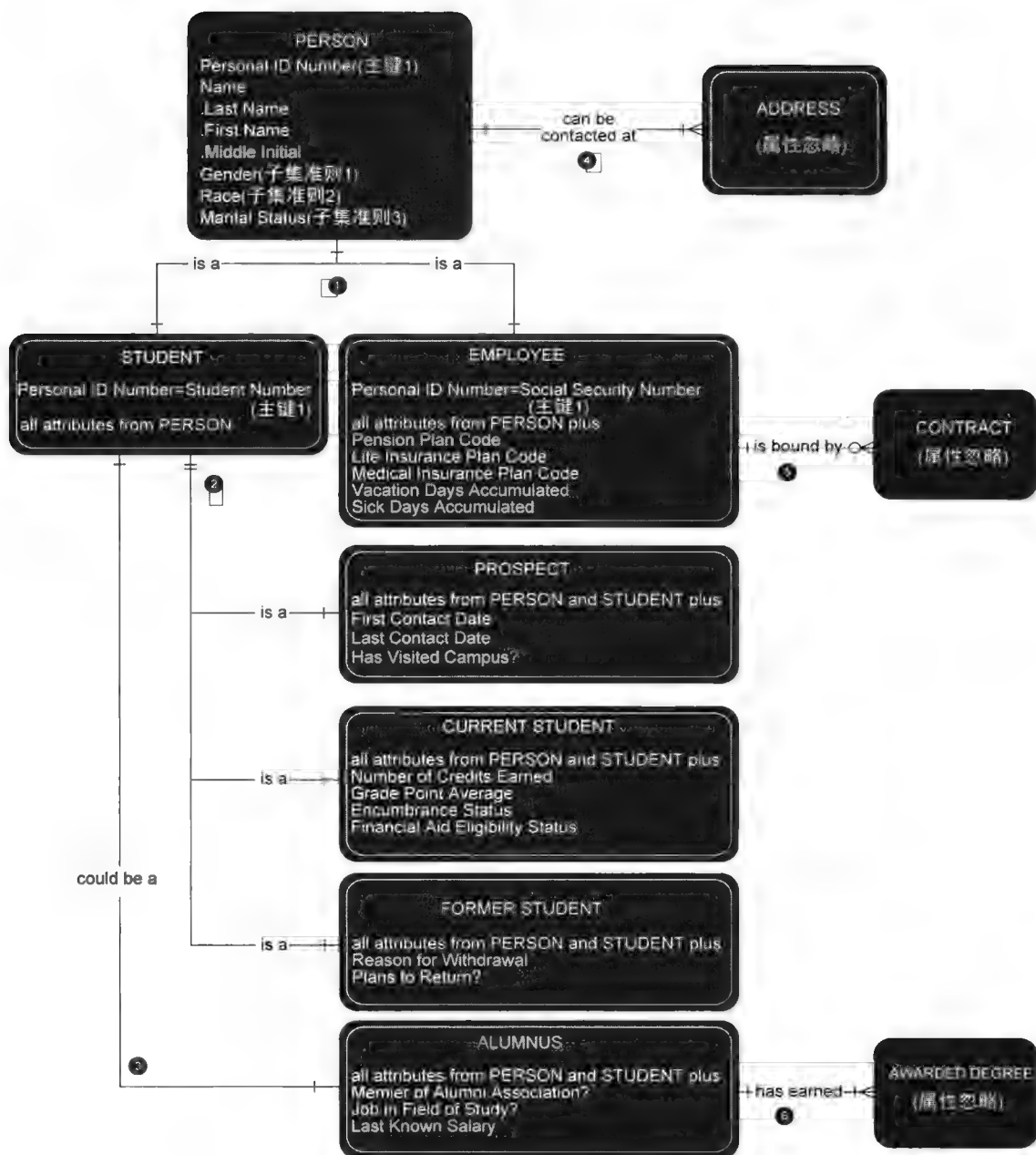


图 7-10 通过识别独立的关系分解非特定关系

7.2.3.4 泛化

大多数人将泛化的概念同现代的面面向对象技术联系起来。实际上，这个概念已经被数据建模人员使用多年了。泛化是一种发现和探索实体之间的共性的方法。泛化是一种技术，其中几类实体公共的属性被组合成实体。例如，考虑一个典型的学校。一个学校登记学生（STUDENT）并雇用雇员（EMPLOYEE）（在大学中，一个人可能同时扮演两个角色）。对这两个实体来说，有几个属性是公共的，例如 NAME、GENDER、RACE、MARITAL STATUS，甚至可能还包括一个基于 SOCIAL SECURITY NUMBER 的键。我们可以提取这些公共属性到一个称为 PERSON 的实体超类中。实体超类是一个实体，其实例存储了一个或多个实体子类的公共属性。

实体超类将拥有一个或多个到实体子类的一对一关系。这些关系有时称为“is a”关系（或“was a”或“could be a”关系），因为超类的每个实例“也是”（is also an）一个或多个子类的实例。实体子类是一个实体，其实例从一个实体超类中继承了一些公共属性，然后增加了其他一些子类实例的独特属性。在我们的例子中，“一个人是一个（is an）雇员，或者是一个学生，或者二者都是。”图 7-11 的上半部分展示了这种泛化过程。注意子类 STUDENT 和 EMPLOYEE 从 PERSON 超类中继承了属性，还增加了各自的属性。



扩展这个概念，一个实体既可以是一个超类也可以是一个子类。回到图 7-11，我们看到一个 STUDENT 实体（它是 PERSON 实体的一个子类）具有它自己的子类。在该图中，我们看到一个学生或者是一个预科生（PROSPECT），或者是一个在册学生（CURRENT STUDENT），或者是一个以前的学生（FORMER STUDENT）（由于某些原因离开而没有毕业），而且一个学生可以是一个校友（ALUMNUS）。这些额外的子类从实体 STUDENT 和 PERSON 继承了所有属性。最后，注意所有子类都是弱实体。

通过继承，数据模型中的泛化概念使得我们可以通过共享公共属性减少属性数量。子类不仅继承了属性，而且继承了那些属性的数据类型、域和默认值。这可以极大地提高应用于许多不同实体的属性的一致性（例如，日期、名字、地址、货币等）。

除了继承属性以外，子类还继承了到其他实体的关系。例如，所有的 EMPLOYEE 和 STUDENT 实体继承了 PERSON 和 ADDRESS 实体之间的关系，但只有 EMPLOYEE 实体继承了 CONTRACT 实体的关系。而且只有一个 ALUMNUS 实体实例可以关联到一个 AWARDED DEGREE 实体实例。

7.3 逻辑数据建模过程

现在你理解了数据模型的基本概念，下面就来介绍数据建模过程。何时进行数据建模？可能要绘制多少数据模型？有什么技术可以支持这个过程？

数据建模可以在各种类型的项目中进行，也可以在项目的多个阶段中进行。数据模型是不断累进的，对于一个企业或应用来说，不存在“最终的”数据模型这种东西。相反，一个数据模型应该被看作是一个活的文档，它将随着需求的变化而变化。理想情况下，数据模型应该存储在资料库中，以便可以随时访问、扩充和编辑。下面介绍在系统计划和分析期间如何进行数据建模。

7.3.1 战略数据建模

许多组织根据战略信息系统规划选择应用开发项目。战略规划是一个独立的项目，这个项目生成了一个信息系统战略规划，它定义一个用于信息系统的整体构想和架构。这个架构几乎总是包括一个企业数据模型。信息工程是一种包含了这种方法的方法学。

企业数据模型一般只是标识出了最基本的实体。这些实体被一般性地定义（如在一个字典中），但它们不会以键或属性的形式进行描述。企业数据模型可能包括也可能不包括关系（取决于规划方法学的标准和主要管理人员期望的详细程度）。如果包括了关系，许多关系都将是非特定关系（在本章前面介绍的一个概念）。

企业数据模型通常存储在公司的资料库中。当应用开发项目启动时，企业数据模型（以及其他模型）的子集就从公司资料库中输出到一个项目资料库中。一旦项目团队完成了系统分析和设计，这个被扩充并被精炼的数据模型就又放回到公司资料库中。

7.3.2 系统分析期间的数据建模

在系统分析部分和本章中，我们将集中讨论作为系统分析一部分的逻辑数据建模。单个信息系统的数据库模型通常称为应用数据模型。

数据模型很少在系统分析的范围定义阶段构造，这个阶段在较短的工作时间内进行建模并不现实。如果存在一个企业数据模型，企业数据模型的一个子集就可以应用于该项目中，可以作为这个阶段确立上下文需求的一部分进行访问和检查。或者，项目团队可以确定一个简单的实体清单，列出关于团队成员认为系统将要收集和存储数据的实体。

问题分析阶段的模型应该仅仅包括实体和关系，而不包括属性，这称为上下文数据模型。其目的是提炼我们对项目范围的理解，而不是获取实体和业务规则的细节，其中许多关系都可以是非特定关系。

需求分析得出一个逻辑数据模型，这个模型按照以下步骤开发：

1. 我们从构造上下文数据模型开始确立项目范围。如果上下文数据模型在系统分析期间已经开发出来，那么可以修改模型以反映新的需求和新的项目范围。

2. 绘制一个基于键的数据模型。这个模型将消除非特定关系，增加关联实体，并包括进主键和替代键。这个基于键的数据模型还将包括精确的基数和泛化层次。

3. 构造一个具有完整属性的数据模型。这个属性完整的数据模型包含了所有描述性属性和子集准则。每个属性都用数据类型、域和默认值在资料库中定义（因此，它有时也称为完全描述的数据模型）。

4. 通过一个称为规范化的过程分析数据模型的适应性和灵活性。最终经过分析后的模型被称为规范化的数据模型。

为了获得这个需求分析模型，需要团队的努力才能完成，其中包括系统分析员、用户和管理人员以及数据分析师的努力。数据管理员经常为所有的数据模型建立标准，并认可数据模型。

最后，没有用户团体提供的合适事实和信息，数据模型是无法构造出来的。这些事实可以通过一些技术进行收集，例如抽样现有表单和文件、研究类似系统、用户和管理人员的概述、与用户和管理人员的面谈。为了收集事实和信息并同时构造和验证数据模型，最快的方法是联合需求计划。JRP 通过小组会议来收集事实、构造模型并验证模型——通常是在一到两天的全天会议中完成。调查研究和信息收集技术在第5章已做了详细介绍。表7-4 总结了一些可能对有关数据建模的调查研究和信息收集技术有用的问题。

表 7-4 用于数据建模的 JRP 和面谈问题

目 的	候 选 问 题
获取系统实体	业务的主体是什么？即什么类型的人、组织、组织部门、地点、事物、材料或事件用于这个系统中，或同这个系统交互（我们需要收集和维持有关这些内容的数据）？每个主体存在多少个实例
获取实体键	哪个唯一特征（或多个特征）将每个主体的实例从同一主体的其他实例中区分开来？未来计划改变这个识别方案吗
获取实体子集准则	一个主体有哪些特征将这个主体的所有实例分成有用的子集？上面的主体是否存在某些子集，而对于这些子集没有方便的方法组织实例
获取属性和域	每个主体用什么特征描述？对于这些特征：1）存储什么类型的数据？2）谁负责为数据定义合法值？3）这个数据的合法值是什么？4）必须有值吗？5）如果没有指定值，是否应该指定默认值
获取安全和控制需求	关于谁可以查看或使用数据有任何限制吗？允许谁创建数据？允许谁修改数据？允许谁删除数据
获取数据时间要求	数据变化有多频繁？在什么时间段内数据对企业有价值？应该保存数据多长时间？需要历史数据或趋势数据吗？如果特征发生变化，必须知道原始值吗
获取泛化层次	每个主体的所有实例都是一样的吗？即每个主体存在需要进行不同地描述或处理的特殊类型吗？数据中有可以提取出来共享的吗
获取关系和度数	什么事件的出现隐含了主体之间的关联？什么业务活动或事务涉及处理和修改几个具有相同类型或不同类型的不同主体的数据
获取基数	每个业务活动或事件是以同样的方式处理，或者存在特殊情况吗？事件能够只同某些相关主体一起出现，或者必须涉及所有主体吗

7.3.3 对系统设计的考虑

在系统设计期间，逻辑数据模型将转变为所选的数据库管理系统的物理数据模型（称为数据库模式）。这个模型将反映数据库技术的技术能力和限制，以及数据库管理员建议的性能优化需求。数据库设计的进一步讨论将在第13章展开。

7.3.4 数据建模的自动化工具

数据模型存储在资料库中。在某种意义上，数据模型是元数据，即关于业务数据的数据。计算机辅助系统工程（CASE）技术（见第2章）为存储数据模型及其详细描述提供了资料库。大多数 CASE 产品支持计算机辅助数据建模和数据库设计，有些 CASE 产品（例如 Logic Works 公司的 ERwin）只支持数据建模和数据库设计。CASE 承担了绘制和维护这些模型及其底层细节的任务。

使用一个 CASE 产品，你可以不用纸、笔、橡皮擦和模板就能方便地创建专业的、易读的数据模型。可以方便地修改模型以反映最终用户建议的更正和修改——不必从头开始！而且，大多数 CASE 产品提供了强有力的分析工具，可以检查模型的手工错误、完整性和一致性。有些 CASE 产品甚至可以帮助你分析数据模型的一致性、完整性和灵活性。节省时间和提高质量的潜力是巨大的。

正如前面所提到的，有些 CASE 工具支持将现有文件和数据库结构逆向工程为数据模型。得到一个“物理”数据模型，它们可以被修订和再工程成为一个新的文件或数据库，或者可以被转换成等价的“逻辑”模型。然后逻辑数据模型可以被编辑并正向工程成为一个修改过的物理数据模型，用于后续的文件或数据库实现。

CASE 工具确实也有其不足。不是所有的数据建模规则都被所有的 CASE 产品支持，而且不同的

CASE 工具对于同样的数据建模方法采用了稍微不同的符号记法。所以，很可能某些 CASE 产品会要求公司采用 CASE 产品的方法学中的数据建模符号或方法，以便可以在 CASE 工具的限制范围内工作。

本章下一节中所有的音阶公司案例的数据模型都是用 Popkin 公司的 CASE 工具 System Architect 2001 创建的。对于这个案例研究，我们为读者提供了与打印稿完全一样的输出。我们没有添加颜色，唯一的艺术修饰是提醒读者注意打印输出中特别重要的内容的着重符号。音阶公司数据模型中的所有实体、属性和关系都被自动地分类存储在 System Architect 的项目资料库中（称为百科全书）。图 7-12 展示了用于数据建模的一些 System Architect 屏幕。

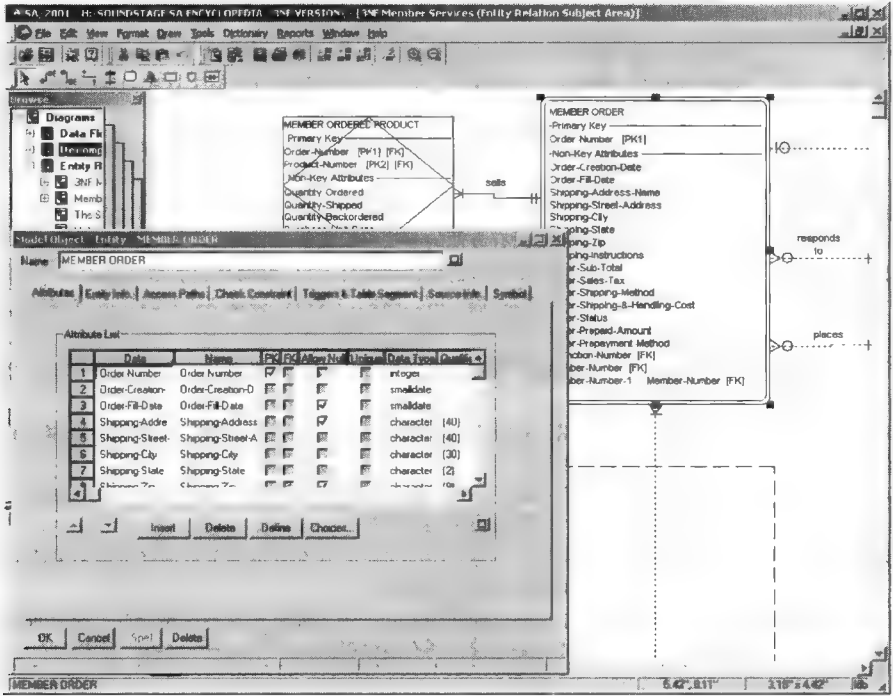


图 7-12 System Architect CASE 工具的屏幕

7.4 如何构造数据模型

你现在了解的数据模型知识已经足够用于阅读和解释模型，但作为一个系统分析员或者有知识的最终用户，你还必须学会如何构造数据模型。我们将使用音阶娱乐俱乐部项目案例教你如何构造数据模型。

注意 该案例教你从头开始绘制数据模型。实际上，一般应该先寻找一个现有模型。如果存在这样的模型，数据管理小组通常会保留它们。或者，可以从一个现有的数据库中逆向工程出一个数据模型。

7.4.1 获取实体

在数据建模中第一项任务相对容易。你需要获取系统中的基本实体，这些基本实体由数据描述，或者可能由数据描述。不要把你对于实体的考虑仅仅限制在最终用户所知范围内。以下几种技术可以用来确定实体：

- 在与系统所有者和用户的面谈或 JRP 会议期间，注意他们讨论中的关键词。例如，在与一个人讨论音阶公司的企业环境和活动时，一位用户可能说道，“我们需要跟踪我们的所有会员及其到期的合同。”注意这句话中的关键词是“会员（MEMBER）”和“合同（AGREEMENT）”，它们都是实体。

- 在面谈或 JRP 会议期间，专门要求系统所有者和用户确定他们想收集、存储和生成信息的事物。那些事物常常表示了应该在数据模型中描述的实体。
- 确定实体的另一个技术是研究现有表格、文件和报告。有些表可以确定事件实体，例子包括 ORDER、REQUISITION、PAYMENT、DEPOSIT 等等。但是这些表格中大多数还包含了描述其他实体的数据。例如考虑你所在学校的课程登记系统中使用的一个注册表，REGISTRATION（注册）本身就是一个事件实体，但一般的注册表也包含了描述其他实体的数据，例如 STUDENT（一个人）、COURSE（概念）、INSTRUCTOR（另一个人）、ADVISOR（又一个人）、DIVISION（另一个概念）等等。研究计算机注册系统的计算机文件、数据库或输出也可以发现这些实体。
- 如果用例描述在需求分析阶段被记录下来，那么它们就可能成为数据属性和实体的来源。扫描每个用例描述中的名词。每个名词都是潜在的数据属性或实体。你将必须修改获得的名词清单，因为并不是所有的名词都是数据属性或实体。有一些是对用户或其他信息系统的索引；有些是对用户界面而不是数据的索引；有些是清单中其他数据属性或实体的同义词，而你并不想重复它们。第 10 章解释如何做这件工作，采用面向对象的方法来创建对象清单及其属性。你可以使用十分类似的方法来发现数据实体及其属性。
- 工具也可以帮助你确定实体。有些 CASE 工具可以逆向工程现有文件和数据库成为物理数据模型。分析员通常必须整理得到的模型，将物理名称、代码和注解替换成业务相关的逻辑等价物。

当实体被发现时，为它们设置简单的、有意义的、面向业务的名字。应该用描述有关存储数据的人、事件、地点、对象或事物的名词来命名实体。尽量不要简写或者使用缩写词。名字应该是单数，以便于区分实体的逻辑概念和实际实例。为了更好地描述实体，名字可以包含合适的形容词或从句。例如，一个外部产生的 CUSTOMER ORDER 实体必须与一个内部产生的 STOCK ORDER 实体区分开来。

应按照业务词汇定义实体，不要用技术词汇定义，也不要定义成“关于……的数据”。试试下面这个办法！使用一本英语字典创建一个初步定义，然后以手边的业务定制它。你的实体名称和定义将生成一个业务词汇的初步词汇表，这个词汇表将为你以及未来的分析员和用户多年的服务^①。

音阶公司的管理人员和用户最初确定了表 7-5 中的实体。注意定义是如何用于建立系统词汇表的。

表 7-5 音阶公司项目的基本实体

实体名称	业务定义
AGREEMENT	会员承诺在一定时间内购买一定数量产品的合同。履行了这个合同后，会员就有资格获得奖金，奖金可以兑换免费或打折的产品
MEMBER	属于一个或多个俱乐部的活跃的会员 注意：最终系统的目标是重新注册不活跃的会员，而不是删除他们
MEMBER ORDER	作为每月促销商品的一部分为会员生成的一份订单，或者由一个会员发出的一份订单 注意：当前的系统仅仅支持从促销商品生成的订单；另外，对客户发出的订单给予了很高的重视，作为建议的系统中的一个新增加的功能
TRANSACTION	会员服务系统必须响应的一个业务事件
PRODUCT	可用于促销商品和销售给会员的库存产品 注意：系统改进目标包括：1) 与正在为仓库开发的新的条形码系统兼容；2) 适应快速变化的产品组合
PROMOTION	一个每月或每季度举行一次的活动，向会员提供特殊的产品价格

7.4.2 上下文数据模型

数据建模的下一个任务是构造上下文数据模型。上下文数据模型应该包括基本业务实体以及它们之间的自然关系。

关系应该使用动词词组和实体名称的组合命名，形成一个简单的业务语句或推断。有些 CASE 工具

① 以上关于实体命名的讨论是基于英语语言环境的，仅供读者参考。——译者注

(例如 System Architect) 允许在两个方向上命名关系。否则, 一般从父实体向子实体的方向命名关系。

我们已经在图 7-13 中完成了这个任务。这张图表示了一个在 System Architect 中构造的数据模型。一旦我们开始映射属性, 新的实体和关系就可能出现。下面的数字注释对应图 7-13 中同样的数字符号。ERD 表达了以下内容:

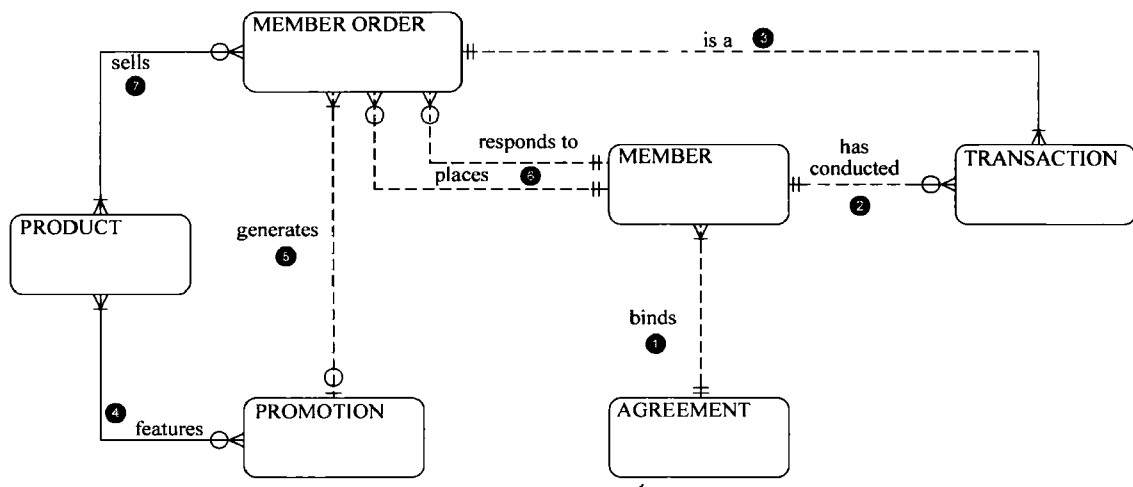


图 7-13 音阶公司案例的上下文数据模型

- ①一份合同绑定 (bind) 了一个或多个会员。虽然关系可以只在一个方向上 (从父到子) 命名, 但另一个方向的命名是隐含的。例如, 这里隐含了一个会员被绑定到一个且仅一个合同。
 - ②一个会员执行 (has conducted) 零个、一个或多个事务 (TRANSACTION)。隐含地, 一个给定的事务被一个且仅一个会员执行。
 - ③一个会员订单 (MEMBER ORDER) 是一个 (is a) 事务。事实上, 一个给定的会员订单可以对应到许多事务 (例如, 一个新会员订单, 一个取消的会员订单, 一个改变了的会员订单等等)。但一个给定的事务可能 (也可能不) 表示 (represent) 一个会员订单。
 - ④一个促销商品 (PROMOTION) 重点推介 (feature) 了一个或多个产品。隐含地, 一个产品在零个、一个或多个促销商品中重点推介。例如, 一张既对乡村/西部听众有吸引力又对轻松的摇滚听众有吸引力的 CD 可以同时在这二者的促销商品中重点推介。因为产品数量远远多于促销商品数量, 所以大多数产品从来没有在促销商品中重点推介过。
 - ⑤一个促销商品产生 (generate) 多个会员订单。隐含地, 一个会员订单为零个、一个或多个促销商品产生。为什么会出现零个促销商品? 因为在新系统中, 一个会员将能够自己发出他 (或她) 的订单。
 - ⑥如果不同的关系沟通了不同的业务事件或关联, 那么同样的两个实体之间也允许存在多个关系。这样, 一个会员响应 (respond to) 零个、一个或多个会员订单, 这个关系支持了促销商品产生的订单。一个会员发出 (place) 零个、一个或多个会员订单, 这个关系支持了会员发出的订单。在这两种情况下, 一个会员订单完全由一个会员发出 (响应)。
- 尽管我们不需要这种双重关系, 但某些 CASE 工具 (包括 System Architect) 为记录布尔关系 (例如 AND、OR) 提供了专门的符号。这样, 对于任意的两种关系, 可以使用一个布尔符号表示关系的实例之间是必须互斥 (OR) 还是必须互补 (AND)。
- ⑦一个会员订单销售 (sell) 了一个或多个产品。隐含地, 一个产品在零个、一个或多个会员订单中销售。注意这是一个非特定关系, 将在后面被消解。

如果你仔细地阅读了前面的每一项内容, 你可能会对音阶公司系统有深入的了解。作为一种为系统项目描述业务上下文的工具, 数据模型已经越来越流行了。

7.4.3 基于键的数据模型

下一个任务是确定每个实体的键。对于键我们建议遵循如下原则^①：

1. 在每个实体实例的生命期中，一个键的值不应该改变。例如，NAME 属性就是一个不好的键，因为一个人的姓可能会由于结婚或离异而改变。

2. 键的值不能为空。

3. 必须进行控制以确保键的值是有效值。这可以通过精确地定义域，并使用数据库管理系统的验证控制机制强制该域来实现。

4. 有些专家（例如 Bruce）建议不要使用智能键。一个智能键是一个业务编码，其结构表达了一个实体实例的数据（例如它的分类、尺寸或其他属性）。一个编码是一组标识和描述业务系统中的某些事物的字符和/或数字。他们认为由于那些字符可能会变化，所以违反了上面的第1条原则。

我们不赞成这一建议，我们认为业务编码可以给组织带来价值，因为人们可以在没有计算机辅助下快速地处理它。

a. 编码有几种类型。它们可以组合起来有效地标识实体实例。

1) 序列码给实体实例指定顺序生成的数字。许多数据管理系统可以按照某个业务需求产生并限制序列码。

2) 块状编码类似于序列码，差别是数字块被分成有某些业务含义的组。例如，一个卫星电视供应商可以指定频道 100 ~ 199 作为按次付费频道，频道 200 ~ 299 作为有线电视频道，频道 300 ~ 399 作为体育频道，频道 400 ~ 499 作为成人节目频道，频道 500 ~ 599 作为音乐频道，频道 600 ~ 699 作为交互式游戏频道，频道 700 ~ 799 分配给因特网频道，频道 800 ~ 899 分配给需要额外付费的有线电视频道，频道 900 ~ 999 分配给需要额外付费的电影和新闻频道。

3) 字母编码使用字母（也可能是数字）的有限组合来描述实体实例。例如，每个 STATE 属性都有一个唯一的两位字母编码。字母编码通常必须同序列码或块状编码一起使用，以唯一地标识大多数实体实例。

4) 在显著位置编码中，每个数字或数字组描述了实体实例的一个可度量或可确定的特征。显著位置编码经常用来对库存项目进行编码。你在轮胎和灯泡上看到的编码就是显著位置编码的例子。它们告诉了我们许多特征，例如轮胎尺寸和功率。

5) 层次编码为一个实体实例提供了一种自顶向下的解释。每个被编码的项目都被分解成组、子组等等。例如，我们可以按照如下方式对雇员职位进行编码：

- 第1个数字标识分类（例如，办事员、教员等）。
- 第2个和第3个数字指示分类中的等级。
- 第4个和第5个数字指示雇用年限。

b. 当创建一个业务编码方案时，建议采用以下指南：

- 1) 编码应该可扩展。
- 2) 全部编码应该对每个实体实例指定一个唯一的值。
- 3) 编码应该足够大以描述有区别的特征，但又要足够小以可以被没有计算机的人解释。
- 4) 编码应该方便使用，新的实例应该容易创建。

5. 考虑使用一个代理键来代替独立实体中的大型复合键。这条建议对于关联实体不切实际，因为复合键的每个部分都是一个必须精确地匹配其父实体的主键的外部键。

图 7-14 是音阶公司项目的基于键的数据模型，注意其中为每个实体指定的主键。

①许多实体都有一个简单的单一属性作为主键。

① 摘自 Thomas A. Bruce, *Designing Quality Databases with IDEFIX Information Models* Copyright © 1992 by Thomas A. Bruce. Reprinted by permission of Dorset House Publishing, 353 W. 12th St., New York, NY 10014 (212-620-4053/1-800-DH-BOOKS/www.dorsethouse.com). All rights reserved.

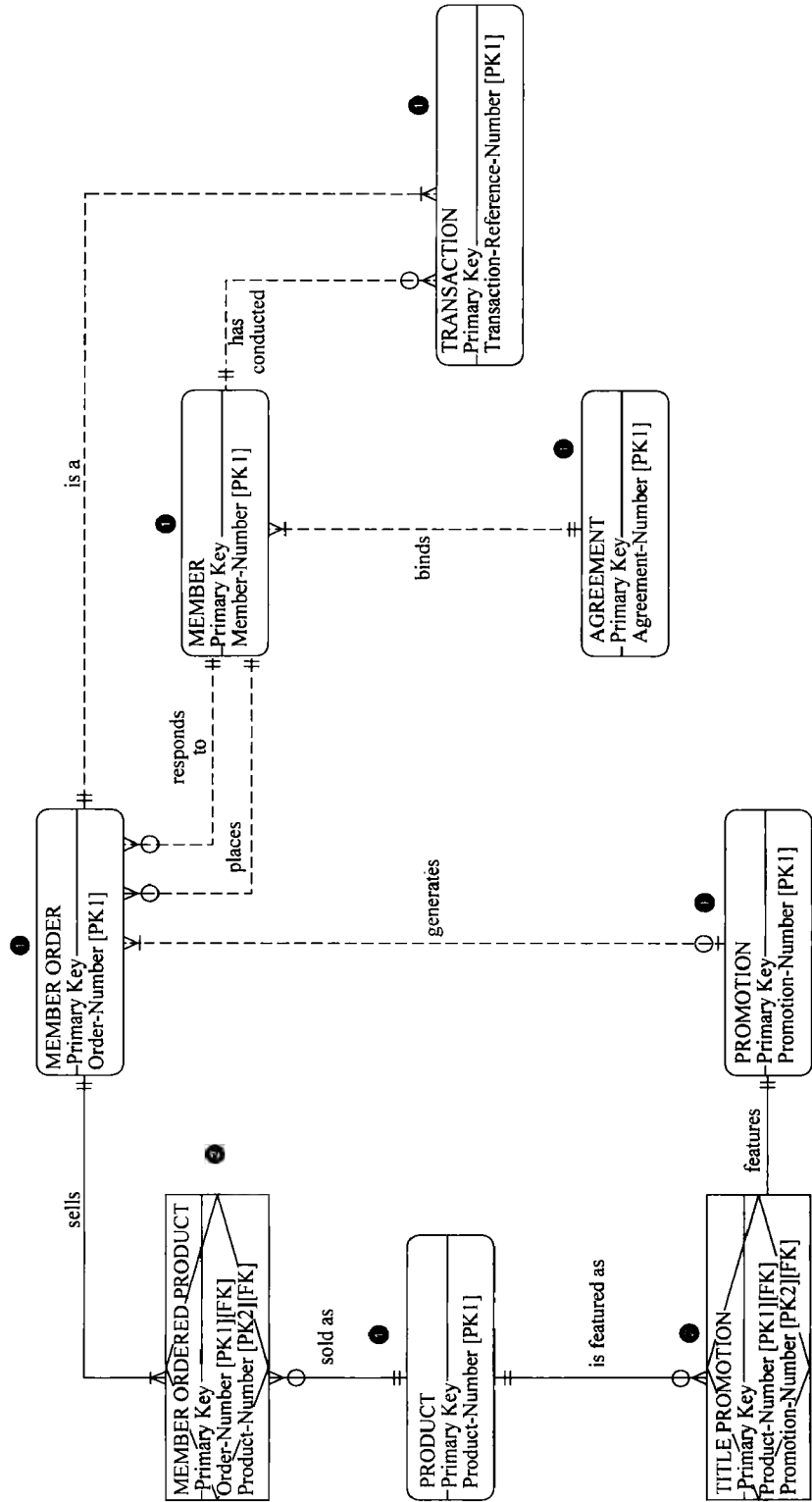


图 7-14 音阶公司案例的基于键的数据模型

- ② 我们通过引入关联实体 MEMBER ORDERED PRODUCT 分解了 MEMBER ORDER 实体和 PRODUCT 实体之间的非特定关系。每个关联实体实例都代表了一个会员订单中的一个产品。父实体贡献了其主键以构成关联实体的复合键。System Architect 在靠近 ORDER NUMBER 属性的地方标示了一个“PK1”，指示它是复合键的“第一部分”，而在 PRODUCT NUMBER 属性边上标示了一个“PK2”，指示它是复合键的“第二部分”。还要注意在该复合键中的每个属性又是一个指回到对应的父实体实例的外部键。

开发这个模型时需注意两点：如果你不能为一个实体定义键，则很可能是这个实体并非真正存在，即这个所谓实体的多次出现并不存在。因此，在全面地定义数据模型的属性之前，指定键是一个不错的质量检查。另外，如果两个或多个实体具有同样的键，它们就很可能是同一实体。

7.4.4 泛化层次体系

现在，需要确定业务领域中的泛化层次体系。音阶公司项目在本章开始时就确定了至少一个超类/子类结构，后面的讨论确实揭示了一个泛化层次体系。这样，我们对基于键的数据模型进行了修正，如图 7-15 所示。由于层次体系的原因，我们必须将模型布局得有些不同，但是仍然保留以前定义的关系和键。读者要注意以下注释：

- ① 音阶公司的 CASE 工具自动地围绕泛化层次体系绘制了一个细线框。
- ② 子类继承了超类的键。
- ③ 如前所示，我们把 PROMOTION 实体从 PRODUCT 实体断开，并重新连接它到子类 TITLE 上。这样做符合业务规则：商品（MERCHANDISE）从不会被一个促销商品（PROMOTION）重点推介——只有名称（TITLE）会被重点推介。

7.4.5 具有完整属性的数据模型

看上去确定其余的数据属性是一项琐碎的任务，但是不熟悉数据建模技术的分析员经常会遇到麻烦。为了完成这项任务，你必须全面地理解系统的数据属性，这些事实可以使用自顶向下的方法（例如集体讨论）或者自底向上的方法（例如抽样表和文件）来收集。如果存在一个企业数据模型，一些（也许很多）属性就可能已经被确定并记录在资料库中了。

对于确定属性的工作，我们提供以下指南：

- 许多组织拥有命名标准和认可的简写方式。由数据管理员维护这个标准。
- 仔细地选择属性的名称。许多属性共享了公共的基本名称，例如 NAME、ADDRESS、DATE。除非属性可以被泛化成为一个超类，否则最好给每个变量一个唯一的名称，例如：

CUSTOMER NAME	CUSTOMER ADDRESS	ORDER DATE
SUPPLIER NAME	SUPPLIER ADDRESS	INVOICE DATE
EMPLOYEE NAME	EMPLOYEE ADDRESS	FLIGHT DATE

另外请记住，无论过去或者将来，一个项目都不是孤立于其他项目而独立存在的。对不同项目的名字应能够多加以区分。

有些组织对那些公共基本属性保留了可复用的全局模板。这倡导了所有应用系统都使用一致的数据类型、域和默认值。

- 现有表格和文件中的物理属性名称经常被简写以节省空间。逻辑属性名称应该更清晰些，例如，转换订单表格中的属性 COD 成为其逻辑等价 AMOUNT TO COLLECT ON DELIVERY，转换属性 QTY 成为 QUANTITY ORDERED 等等。
- 许多属性只可取值 YES 或 NO，试着以问题的形式命名这些属性。例如，属性名称 CANDIDATE FOR A DEGREE 的值是 YES 和 NO。

每个属性都应该仅映射到一个实体。如果一个属性真的描述了不同实体，它可能是几个不同的属性。为每个属性命名一个唯一的名字。

- 外键是对非冗余规则的一个例外——它们确定了相关实体的关联实例。

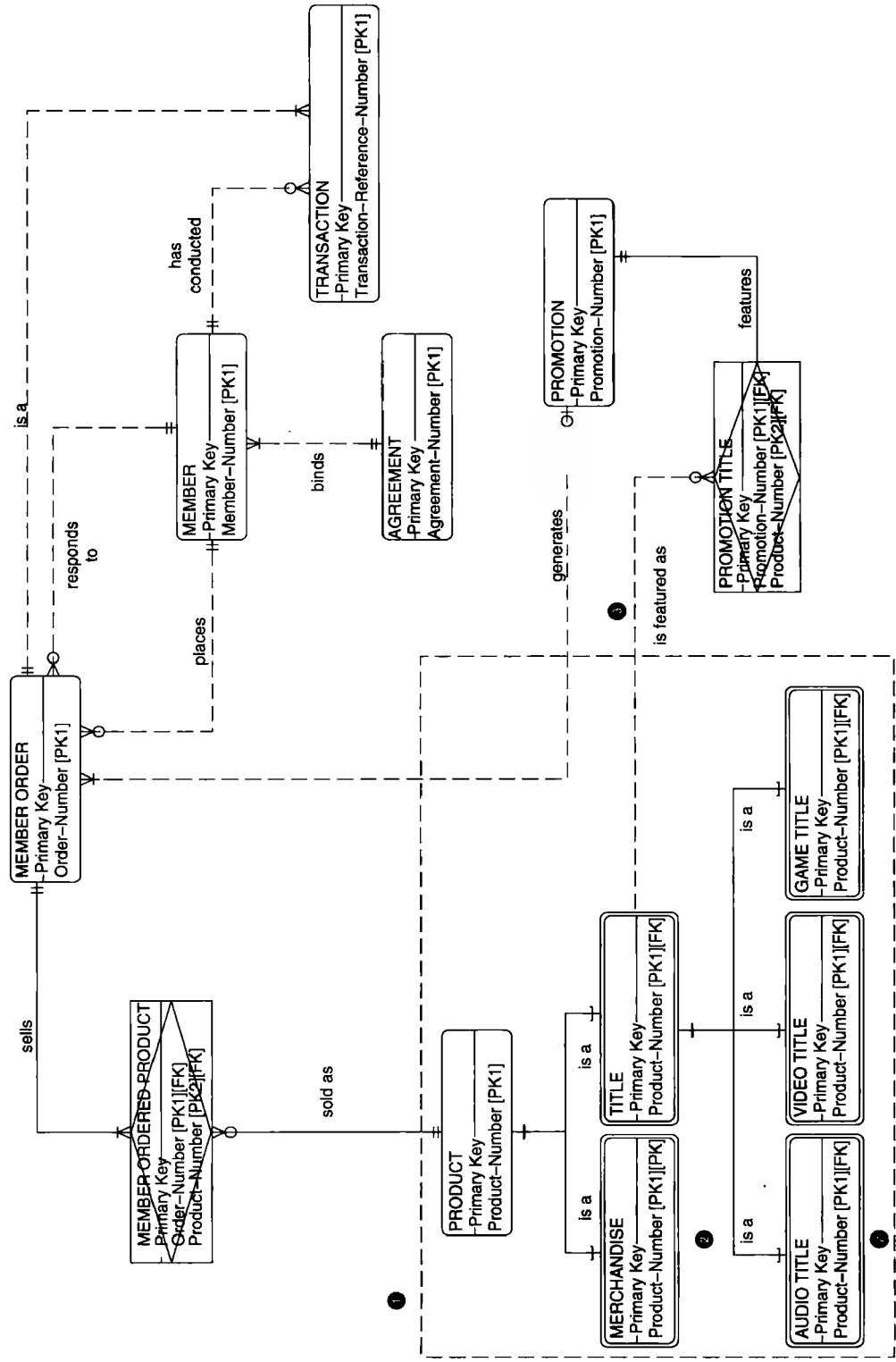


图 7-15 音阶公司案例的含有一个泛化层次体系的基于键的数据模型

- 一个属性的域不应该是逻辑的。例如，在音阶公司案例中，我们知道 MEDIA 属性的值依赖于产品类型。如果产品类型是视频，则介质可能是 VHS 磁带、8mm 磁带、激光碟或 DVD；如果产品类型是音频，则介质可能是卡式磁带、CD 或者 MD。最好的方案是为每个域指定独立的属性：AUDIO MEDIA 和 VIDEO MEDIA。

图 7-16 提供了音阶公司系统项目定义阶段的数据属性到实体的映射。虽然具有完整属性的数据模型确定了在我们未来的数据库中要收集和存储的所有属性，但对那些属性的描述还是不完全的——我们还需要描述域。大多数 CASE 工具提供了大量的工具用于在资料库中描述所有属性的数据类型、域和默认值。另外，应该详细定义每个属性以供日后参考。

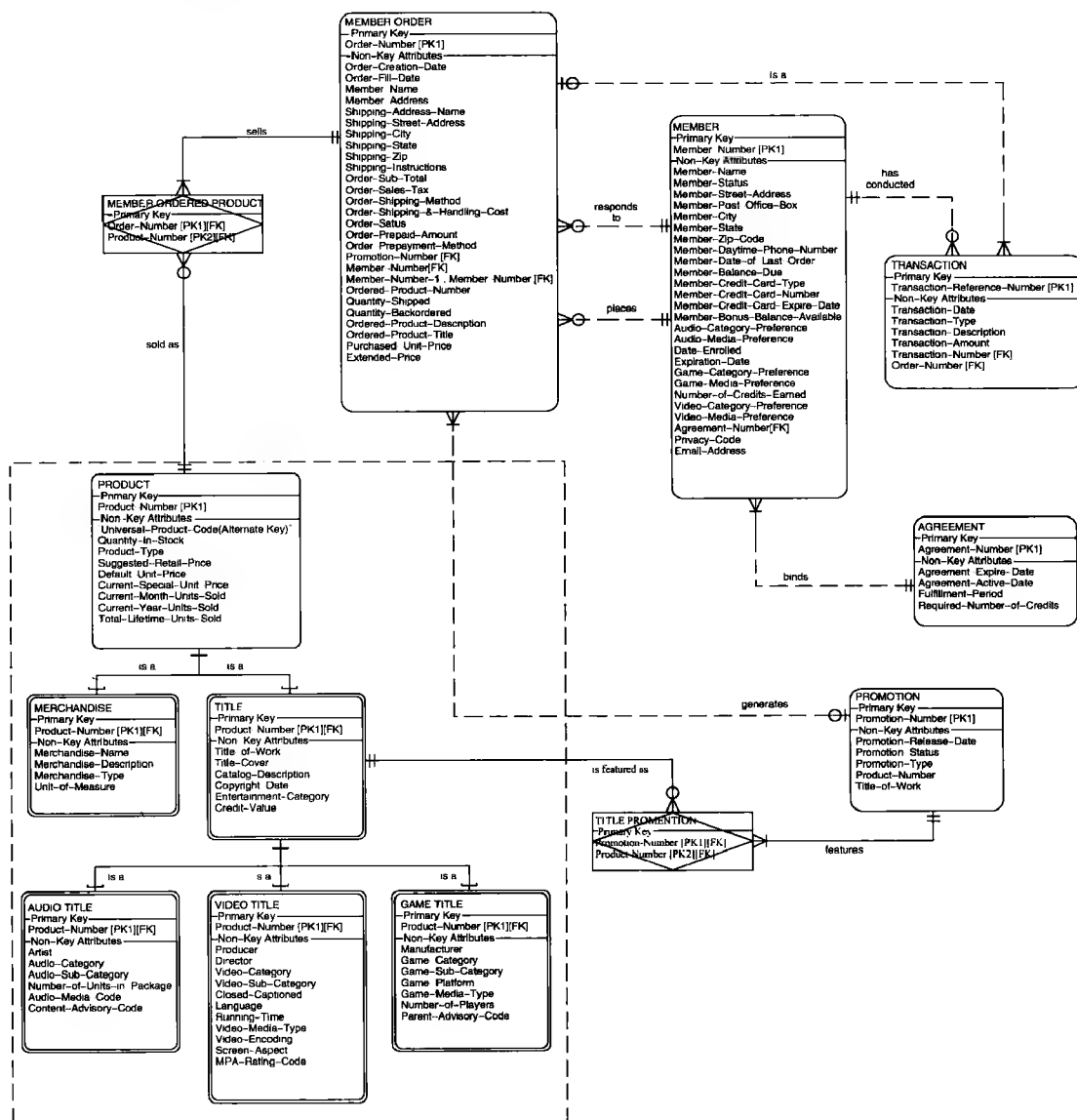


图 7-16 音阶公司的具有完整属性的数据模型

7.5 分析数据模型

虽然数据模型有效地沟通了数据库需求，但它不一定就代表了一个好的数据库设计，其中的某些结构特征可能会降低模型的灵活性和扩展性，或者产生不必要的冗余。所以，我们必须为数据库设计

和实现准备好具有完整属性的数据模型。

本节将讨论一个高质量的数据模型的特征——使我们能够开发出一个理想的数据库结构的数据模型。我们还将介绍在设计数据库之前用来分析数据模型质量并做出必要修改的过程。

7.5.1 好的数据模型的标准

什么是好的数据模型？本书建议采用以下的评价准则：

- 好的数据模型是简单的。作为一条通用规则，描述任何给定实体的数据属性应该仅仅描述那个实体。例如，考虑以下的实体定义：

COURSE REGISTRATION = COURSE REGISTRATION NUMBER (主键) +
 COURSE REGISTRATION DATE +
 STUDENT ID NUMBER (外键) +
 STUDENT NAME +
 STUDENT MAJOR +
 一个或多个 COURSE NUMBER

属性 STUDENT NAME 和 STUDENT MAJOR 真的描述了课程注册的一个实例吗？或者它们仅仅描述了一个不同的实体，即 STUDENT 同样的理由可以应用到属性 STUDENT ID NUMBER 上。经过进一步的考察，STUDENT ID NUMBER 属性仍是需要的，它用来“指出”对应的 STUDENT 实体实例。简单性的另一个方面是这样陈述的：一个实体实例的每个属性只能有一个值。再看一下上面的例子，对应一个 COURSE REGISTRATION 实体，属性 COURSE NUMBER 可以有多个值供学生选择。

- 好的数据模型基本上是无冗余的。这意味着每个数据属性（除了外键）最多在一个实体中描述。在前面的例子中，我们发现属性 STUDENT NAME 和 STUDENT MAJOR 实际描述了一个 STUDENT 实体并不困难。我们应该做出选择。根据前面的要点，合乎逻辑的选择将是添加 STUDENT 实体。在数据模型中还可能更细微的冗余，例如：相同的属性可能以不同的名称（同义词）被多次记录。
- 好的数据模型应该是灵活的而且对未来的需求具有可适应性。没有这条评价准则，我们往往会设计仅实现目前业务需求的数据库。然后，当一个新的需求产生时，我们若不重写许多或者所有使用那些数据库的程序就无法方便地修改数据库。虽然我们不能改变这个现实——大多数项目都是应用驱动的，但是我们可以使数据模型做到尽可能地与应用无关，以鼓励采用不影响当前程序扩展或修改的数据库结构。

那么如何实现上述目标呢？如何设计一个可以适应未来无法预测的需求的数据库呢？答案就在数据分析中。

7.5.2 数据分析

在为数据库设计做准备的过程中，用来改进一个数据模型的技术称为数据分析。数据分析是为实现简单的、无冗余的、灵活的并可扩展的数据库而准备数据模型的过程。其专门的技术称为规范化。规范化是一种数据分析技术，该技术组织数据属性以便它们可以组合起来形成无冗余的、稳定的、灵活的并具有适应性的实体。规范化是一种包括三个步骤的技术，该技术把数据模型规范成第一范式、第二范式和第三范式^①。不要对词汇感到迷惑，这些词汇要比听起来容易得多。现在，让我们对这三个范式建立一个初步的理解：

- 简单地说，如果所有属性对于实体的单个实例都只具有一个值，则这个实体是第一范式（1NF）的。任何可以有多个值的属性实际描述了一个独立的实体，也可能是一个实体和关系。
- 如果实体已经是第一范式的，并且如果所有非主键属性的值都依赖于主键——不仅仅是部分地

① 数据库专家已经确定了其他的范式。第二范式消除了大部分的不规则性。我们把高级范式留给数据库课本和数据库课程讨论。

依赖, 则这个实体是第二范式 (2NF) 的。任何仅仅部分地依赖主键的非键属性应该被移到另一个实体中, 在那里部分键实际上是完全键。这可能需要在模型中创建一个新的实体和关系。

- 如果实体已经是第二范式的, 并且如果它的非主键属性的值不依赖于任何其他非主键属性, 则这个实体是第三范式 (3NF) 的。任何依赖于其他非键属性的非键属性必须去掉或删除。新的实体和关系可能要被添加到数据模型中。

7.5.3 规范化举例

有很多种方法进行规范化。我们选择介绍一种非理论的并且非数学的方法。我们将把理论、关系代数和详细含义留给数据库课程和数据库课本讲解。

同往常一样, 我们将使用音阶公司的案例研究来说明这些步骤。首先来考察早先开发的具有完整属性的数据模型 (见图 7-16)。它是一个规范化的数据模型吗? 不是。让我们找出其中的问题, 并逐步地介绍规范化数据模型的步骤。

7.5.3.1 第一范式

数据分析的第一步是将每个实体变成 1NF。在图 7-16 中, 哪个实体不是 1NF 的?

你应该可以找到两个实体不是 1NF 的——MEMBER ORDER 和 PROMOTION。每个实体都包含了一个重复的组, 即对实体的单个实例可以有多个值的一组属性 (用大括号表示)。这些属性“作为一组属性”被重复了很多次。例如, 考虑实体 MEMBER ORDER。一个会员订单实体可能包括很多产品, 所以, 属性 ORDERED PRODUCT NUMBER、ORDERED PRODUCT DESCRIPTION、ORDERED PRODUCT TITLE、QUANTITY ORDERED、QUANTITY SHIPPED、QUANTITY BACKORDERED、PURCHASED UNIT PRICE 和 EXTENDED PRICE 可以 (并且可能会) 对 MEMBER ORDER 实体的每个实例重复。

类似地, 因为一个促销商品可能重点推介多个产品名称, 所以 PRODUCT NUMBER 和 TITLE OF WORK 属性可能重复。我们应该如何修改模型中的这些不规范的地方?

图 7-17 和图 7-18 说明了如何将这两个实体转变成 1NF。原来的实体画在图的左边, 1NF 的实体画在图的右边。每张图都演示了规范化如何改变数据模型和属性分配。为了读者阅读方便, 受影响的属性用黑体字和小型大写字母表示。

在图 7-17 中, 我们首先去掉对 MEMBER ORDER 实体的一个实例可能有多个值的属性, 这就使 MEMBER ORDER 实体成为了 1NF。但是如何处理被去掉的属性呢? 我们不能把它们完全地从模型中去掉——它们是业务需求的一部分。所以, 我们把整组属性移到一个新的实体 MEMBER ORDERED PRODUCT 中。这些属性的每个实例都描述了一个会员订单中的一个产品。因此, 如果一个特定的订单包含 5 个产品, MEMBER ORDERED PRODUCT 实体就将有 5 个实例。对于每个属性, 每个实体实例都仅包含一个值, 所以新的实体也是 1NF 的。

1NF 的另一个例子 (PROMOTION 实体) 在图 7-18 中给出。同前面一样, 我们把重复的属性移到另一个实体 TITLE PROMOTION 中。

所有其他实体已经是 1NF 的了, 因为它们不包含任何重复的组。

7.5.3.2 第二范式

数据分析的下一步是把实体变成 2NF, 这需要你已把所有的实体变成了 1NF 的。2NF 寻找其值仅由主键的一部分决定而不是由整个复合键决定的属性。因此, 具有单一属性主键的实体就都已经是 2NF 的了, 这包括了实体 PRODUCT (和它的子类)、MEMBER ORDER、MEMBER、PROMOTION、AGREEMENT 和 TRANSACTION。这样, 我们只需要检查那些具有复合键的实体 (MEMBER ORDERED PRODUCT 和 TITLE PROMOTION)。

首先, 检查 MEMBER ORDERED PRODUCT 实体, 其大部分属性依赖于全部主键。例如, QUANTITY ORDERED 属性将没有任何意义, 除非你同时拥有一个 ORDER NUMBER 和一个 PRODUCT NUMBER 属性值。想一想! ORDER NUMBER 本身是不足以确定订单的, 因为订单可能会有许多的订购量, 就像订单上的产品一样。类似地, PRODUCT NUMBER 属性本身也是不足以确定订单的, 因为相同的产品可以出现在多个订单中。这样, QUANTITY ORDERED 属性需要键的两个部分并且依赖于全部

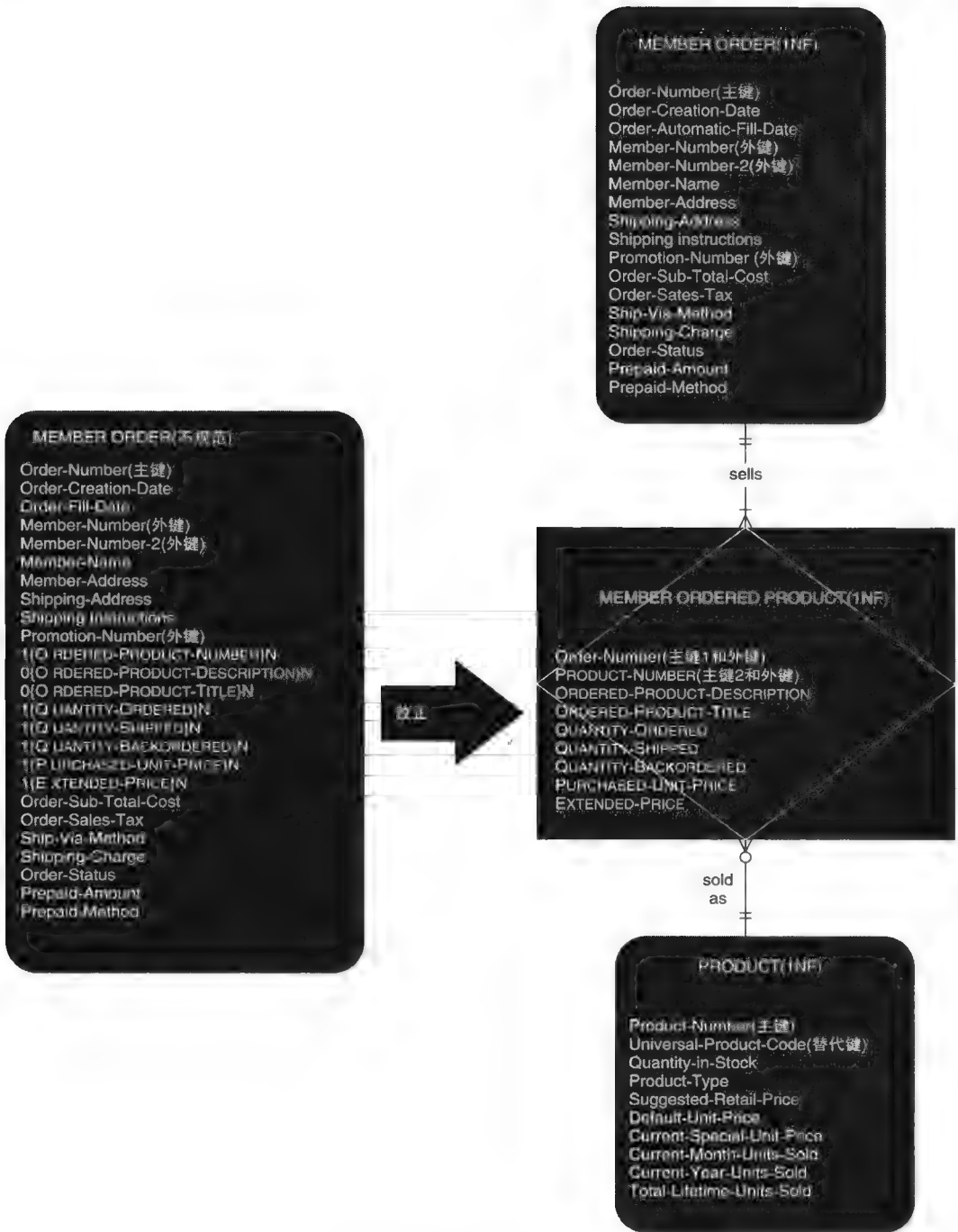


图 7-17 第一范式 (1NF) (一)

键。同样的情况可以应用于属性 QUANTITY SHIPPED、QUANTITY BACKORDERED、PURCHASE UNIT PRICE 和 EXTENDED PRICE。

但是属性 ORDERED PRODUCT DESCRIPTION 和 ORDERED PRODUCT TITLE 怎么样呢？我们真的需要 ORDER NUMBER 属性来确定这两个属性的值吗？不！相反，这些属性的值仅仅依赖于 PRODUCT NUMBER 属性的值。因此，这些属性不依赖于全部键——我们已经发现了一个必须改正的部分依赖不规则性。如何改正这类规范化错误呢？

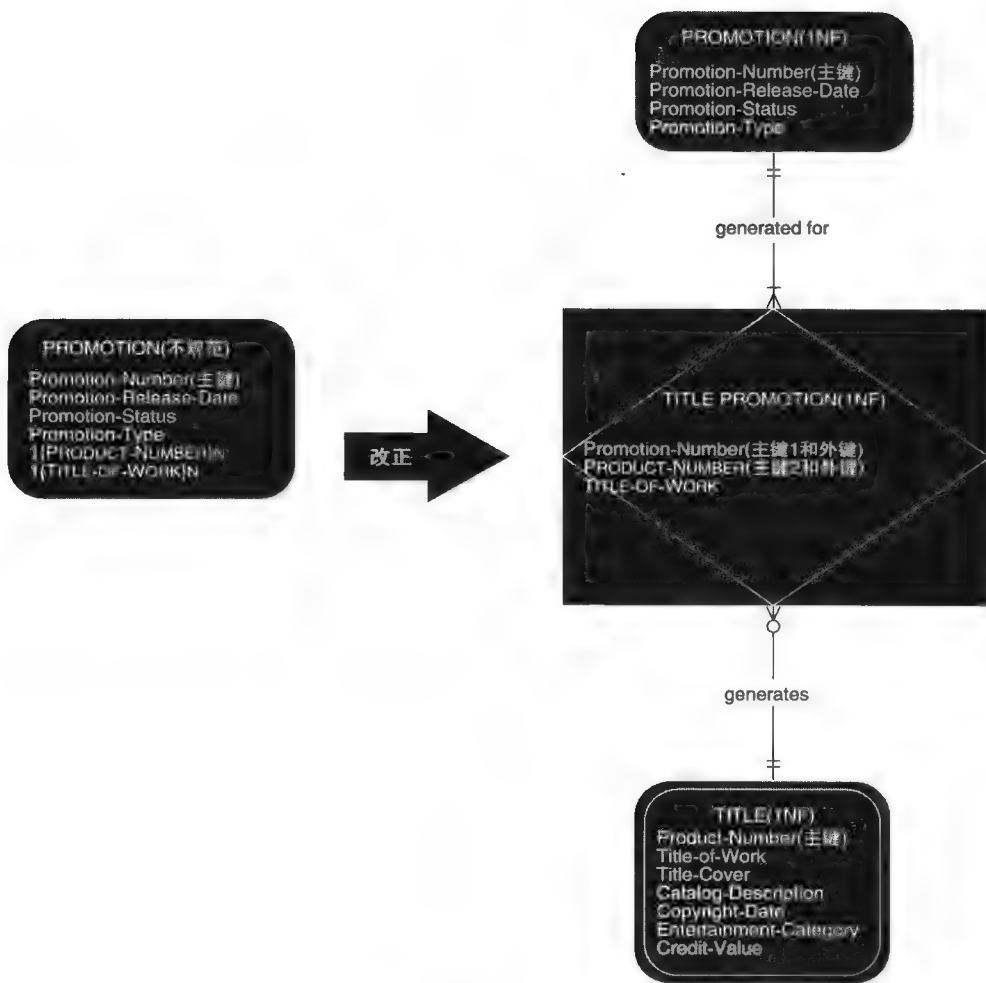


图 7-18 第一范式 (1NF) (二)

参考图 7-19。为了改正这个问题，我们简单地移动非键属性（ORDERED PRODUCT DESCRIPTION 和 ORDERED PRODUCT TITLE）到一个只有 PRODUCT NUMBER 属性作为其键的实体中。如果需要，将需要创建这个实体，但包含那个键的 PRODUCT 实体已经存在了。不过我们要小心，因为 PRODUCT 实体是一个超类。检查了这个超类后，我们发现这些属性已经以同义词的形式存在于 MERCHANDISE 和 TITLE 实体中。因此，实际上我们并不从 MEMBER ORDERED PRODUCT 实体移动属性，只是把它们作为冗余属性删除掉。

下面，让我们检查 TITLE PROMOTION 实体，其复合键是 PROMOTION NUMBER 属性和 PRODUCT NUMBER 属性的组合。TITLE OF WORK 属性依赖于复合键的 PRODUCT NUMBER 部分。因此，TITLE OF WORK 属性就被从 TITLE PROMOTION 中移走（见图 7-20）。注意 TITLE OF WORK 已经存在于实体 TITLE 中，这个实体以一个产品编号作为主键。

7.5.3.3 第三范式

我们可以通过将实体转变成 3NF 进一步简化它们。在开始 3NF 分析之前，实体必须首先是 2NF 的。第三范式分析寻找两类问题：导出的数据和传递依赖关系。在这两种情况中，基本的错误都是非键属性依赖于其他非键属性。

第一类 3NF 分析很容易，检查每个实体的导出属性。导出属性是其值可以从其他属性中计算出来或者可以从其他属性值通过逻辑导出的属性。如果仔细想想，就会发现存储导出属性没有什么意义。

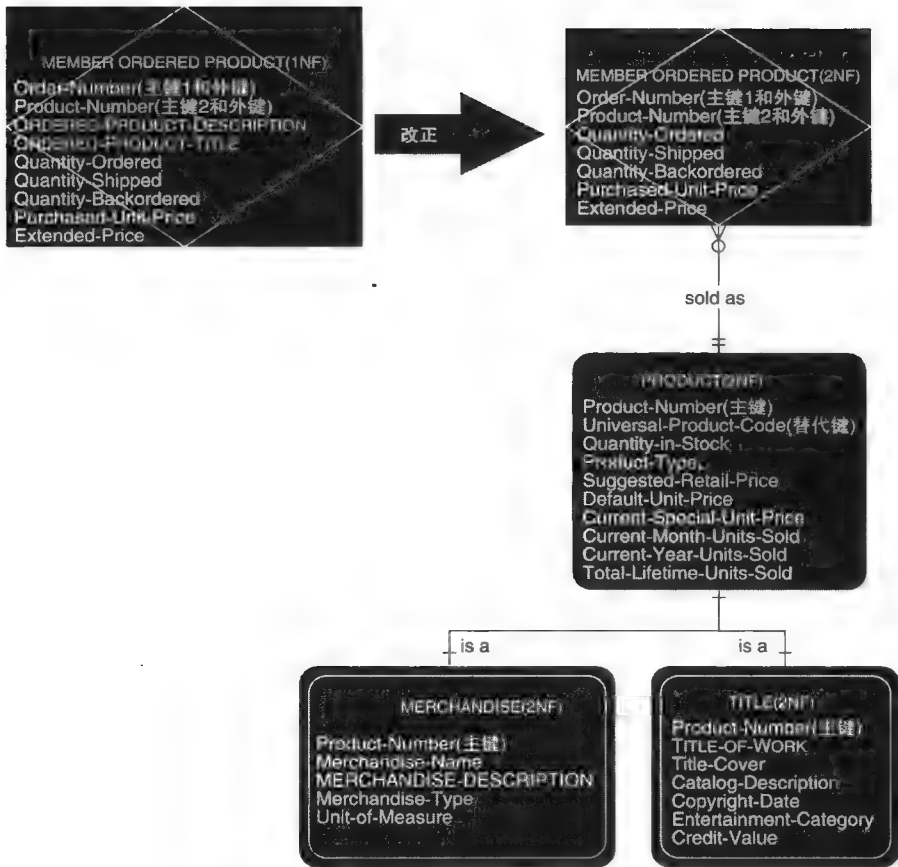


图 7-19 第二范式 (2NF) (一)

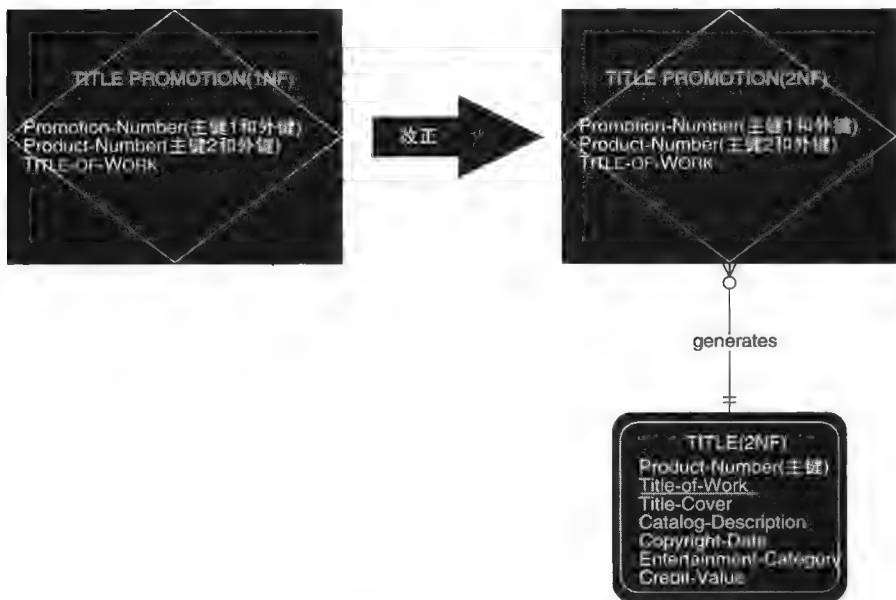


图 7-20 第二范式 (2NF) (二)

首先，它浪费了磁盘存储空间；第二，它使简单的修改变得复杂化。为什么？因为每次修改基本属性时，都必须记着重新计算并修改导出属性。

例如，图 7-21 中的 MEMBER ORDERED PRODUCT 实体。属性 EXTENDED PRICE 是通过 QUANTITY ORDERED 属性乘以 PURCHASED UNIT PRICE 属性计算出来的。这样，属性 EXTENDED PRICE（一个非键属性）对主键的依赖程度并不比其对非主键（QUANTITY ORDERED 和 PURCHASED UNIT PRICE 属性）的依赖程度更高。这样，我们可以通过删除 EXTENDED PRICE 属性改正这个实体。

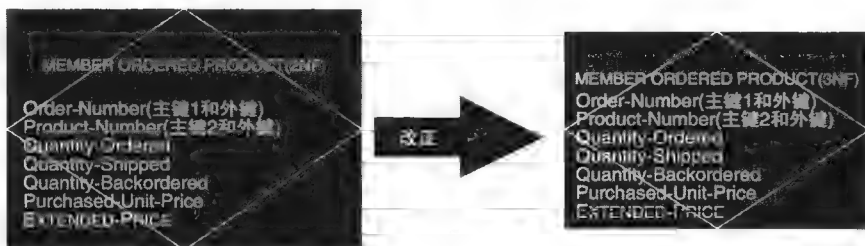


图 7-21 第三范式 (3NF) (一)

听起来很简单，对吗？是的，但并非总是这样简单！对于在多大程度上应用这条规则有不同的意见。有些专家认为这条规则应该仅仅在单个实体的范围内应用。于是，如果导出属性所需的计算是来自不同的实体，这些专家将不会删除它。考虑到涉及多个实体的导出属性会导致更大的数据不一致性危险，我们同意这种看法。这种危险是由于修改一个实体中的属性而忘了接着修改在另一个实体中的导出属性造成的。

另一种形式的 3NF 分析检查传递依赖关系。当一个非键属性依赖于另一个非键属性时（而不是导出），就存在传递依赖关系。这个错误通常指出一个未发现的实体仍然嵌在有问题的实体中。在这种情况下，如果不改正它，当未来的一个新需求最终要求我们把未发现的实体实现成一个独立的数据库表时，就会引起灵活性和适应性问题。

传递分析仅仅在那些没有复合键的实体上进行。在我们的例子中，这包括实体 PRODUCT、MEMBER ORDER、PROMOTION、AGREEMENT、MEMBER 和 TRANSACTION。对于实体 PRODUCT，所有的非键属性都依赖于主键，而且仅仅依赖于主键。这样，PRODUCT 实体就已经是第三范式的了。对 PROMOTION、AGREEMENT 和 TRANSACTION 实体的类似分析表明它们也已经是第三范式的了。

但看看图 7-22 中的实体 MEMBER ORDER。特别地，检查属性 MEMBER NAME 和 MEMBER ADDRESS。这些属性依赖于主键 ORDER NUMBER 吗？不！主键 ORDER NUMBER 无法确定属性 MEMBER NAME 和 MEMBER ADDRESS 的值。另一方面，属性 MEMBER NAME 和 MEMBER ADDRESS 的值依赖于实体中的另一个非主键 MEMBER NUMBER。

如何改正这个问题？属性 MEMBER NAME 和 MEMBER ADDRESS 需要从 MEMBER ORDER 实体中移到一个主键仅仅是 MEMBER NUMBER 的实体中。如果需要，将创建这个实体，但在这个例子中，MEMBER 实体已经具有了所需的主键。因此，不需要真正地移走问题属性，因为它们已经被分配给了 MEMBER 实体。但是，确实注意到 MEMBER ADDRESS 属性是 MEMBER STREET ADDRESS 属性的一个同义词，因此选择在 MEMBER 实体中保留后一个词。

除 3NF 以外还有几种范式。每个更严格的范式都使得模型更加简单、冗余更少、更加灵活。但是，系统分析员（以及大多数数据库专家）很少使数据模型超过 3NF。因此，我们将把有关规范化的进一步讨论留给数据库课本进行。

头几次规范化一个数据模型时，这个过程将显得缓慢而且枯燥。但是，随着时间和实践的增长，这个过程会变得快速而且程序化。许多有经验的建模人员通过在标定属性时就进行规范化（他们能够在开发一个具有完整属性的数据模型的同时进行规范化），从而显著地减少了建模的时间和精力。牢记下面这段话（出处不详）可能会对你有帮助，这段话精妙地总结了第一、第二和第三范式：

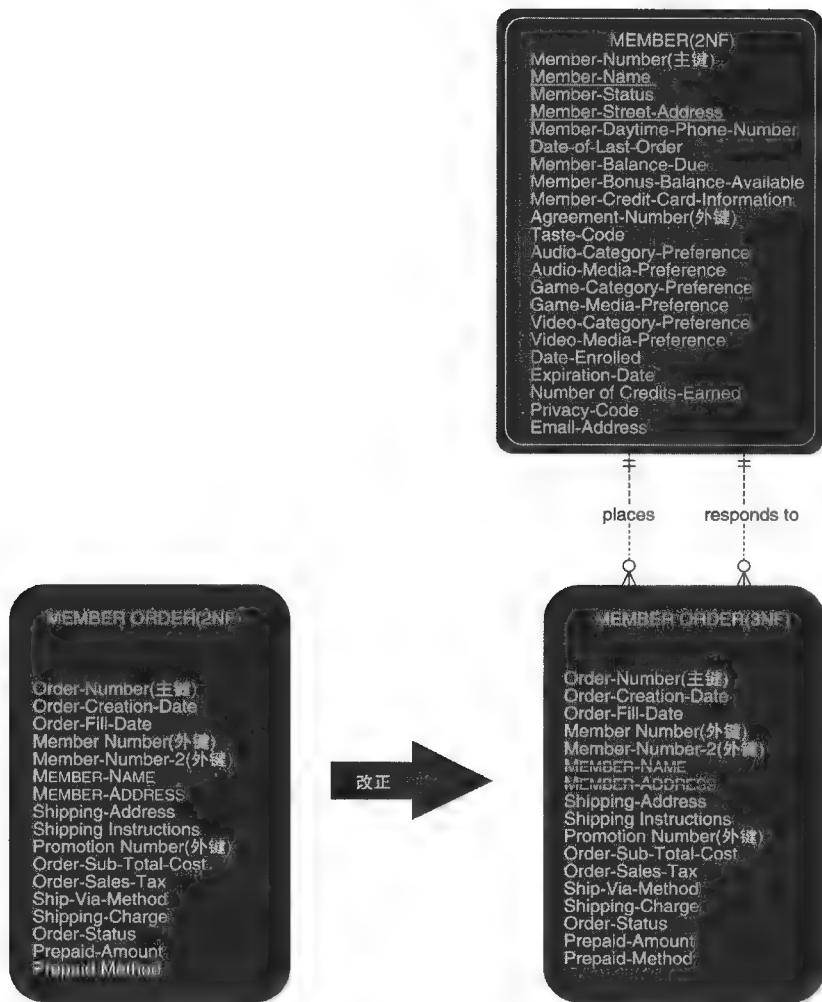


图 7-22 第三范式 (3NF) (二)

如果每个非主键属性都依赖于主键 (整个主键), 而且除了主键以外不再依赖于任何属性, 这个实体就被称为是第三范式的。

7.5.3.4 通过检查进行简化

规范化是一个相当机械的过程, 但它依赖于规范化之前的原始数据模型中的命名一致性。当几个分析员为同一个应用系统工作时, 经常产生规范化以外的问题。这些问题最好通过简化来解决, 这个过程是将 3NF 的数据实体通过进一步去除数据冗余进行简化。

最后的规范化数据模型如图 7-23 所示。

7.5.3.5 对规范化的 CASE 支持

许多 CASE 工具都宣称支持规范化, 它们阅读数据模型并试图孤立可能的规范化错误。但经过仔细的考察后, 你将发现大多数 CASE 工具可能仅仅规范化到了第一范式。它们用两种方法实现这个目标: 寻找多对多关系并把那些关系分解成关联实体, 或者寻找对单个实体具有多个值的属性。有人可能会说分析员应该认识到那是一个 1NF 错误, 而不能这样描述属性。

对于 CASE 工具来说, 要确定第二范式和第三范式错误特别困难。将需要 CASE 工具具有智能功能, 能够识别部分依赖关系和传递依赖关系。在现实中, 这种依赖关系只能通过系统分析员或数据库专家非程序化的检查来发现。

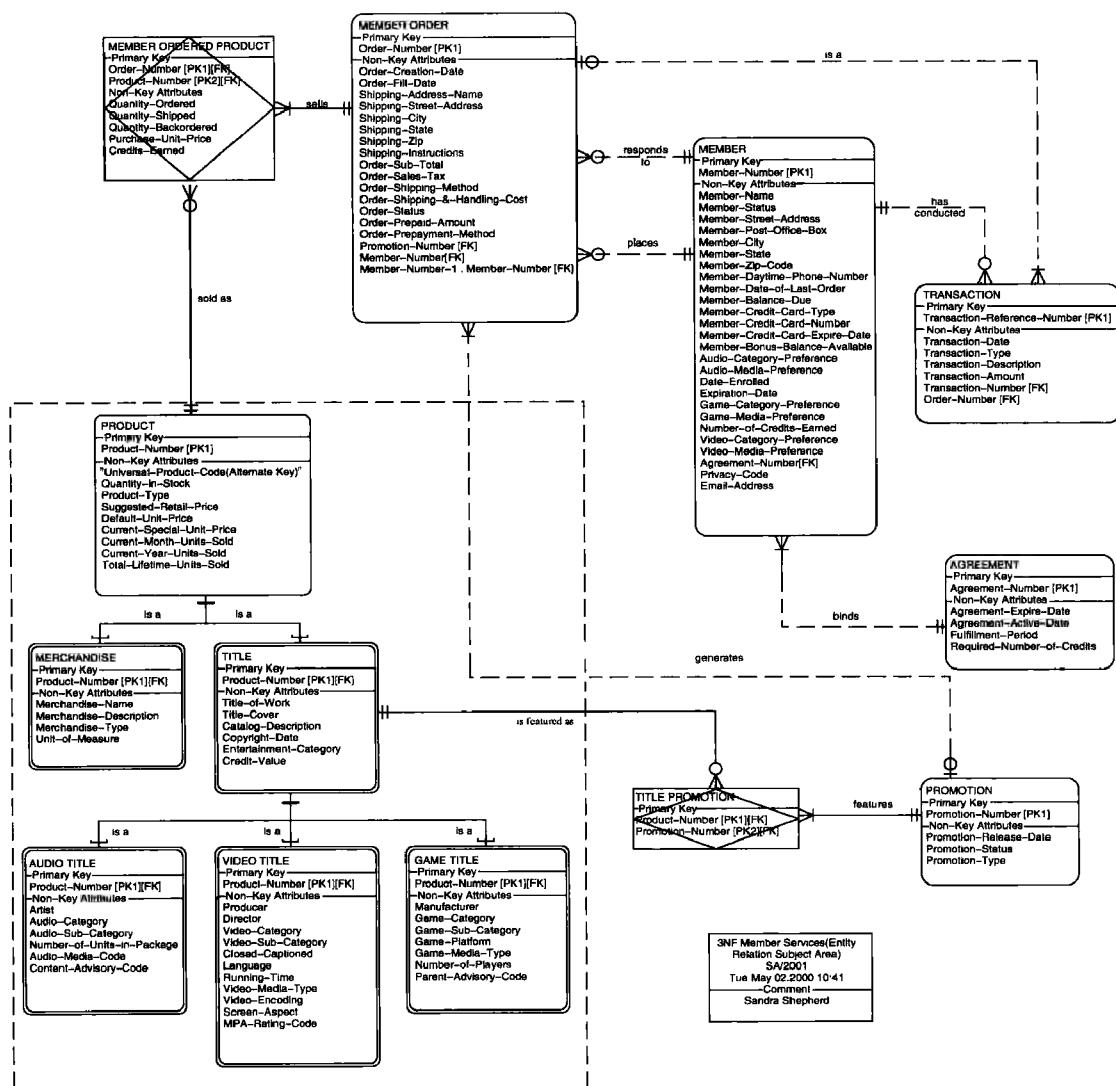


图 7-23 音阶公司的 3NF 的数据模型

7.6 将数据需求映射到地点

虽然一个逻辑数据模型对描述一个新系统需要存储什么数据是有效的，但是它没有沟通有关业务运行地点的需求。需要确定在哪个位置上需要什么样的数据和访问权限。特别地，可以提出以下业务问题：

- 在每个地点需要实体和属性的哪些子集来完成工作？
- 需要什么级别的访问？
- 该地点可以创建实体实例吗？
- 该地点可以读取实体实例吗？
- 该地点可以删除实体实例吗？
- 该地点可以修改实体的现有实例吗？

系统分析员发现按照数据-地点-CRUD 矩阵定义这些逻辑需求是有用的。数据-地点-CRUD 矩阵是一张表，表中行指示了实体（和可能的属性），列指示了地点，单元格（行和列的交叉）记录了访问级别，其中 C = 创建、R = 读取、U = 更新或修改、D = 删除或解除。图 7-24 演示了一个典型的数据-地

点-CRUD 矩阵。是否包括某个属性的决定是根据是否需要限制地点对该属性的访问做出的。图 7-24 也演示了如何记录某个地点需要只对实体实例的某个子集（用 SS 指示）访问的能力。例如，每个销售办公室可能需要只访问其范围内的客户。

Entity.Attribute	Location	Customers	Kansas City	Marketing	Advertising	Warehouse	Sales	APR	Boston	Sales	Warehouse	San Francisco	Sales	San Diego	Warehouse
Customer	INDV						ALL	ALL		SS	SS		SS		SS
Customer Number	R					R	CRUD	R		CRUD	R		CRUD		R
Customer Name	RU					R	CRUD	R		CRUD	R		CRUD		R
Customer Address	RU					R	CRUD	R		CRUD	R		CRUD		R
Customer Credit Rating	X						R	RU		R			R		
Customer Balance Due	R						R	RU		R			R		
Order	INDV			ALL		SS	ALL			SS	SS		SS		SS
Order Number	SRD			R	CRUD	R	CRUD	R		CRUD	R		CRUD		R
Order Date	SRD			R	CRUD	R	CRUD	R		CRUD	R		CRUD		R
Order Amount	SRD			R	CRUD		CRUD	R		CRUD	R		CRUD		R
Ordered Product	INDV			ALL		SS	ALL			SS	SS		SS		SS
Quantity Ordered	SUD			R	CRUD	R	CRUD	R		CRUD			CRUD		
Ordered Item Unit Price	SUD			R	CRUD		CRUD	R		CRUD			CRUD		
Product	ALL			ALL	ALL	ALL	ALL			ALL	ALL		ALL		ALL
Product Number	R			CRUD	R	R	R			R	R		R		R
Product Name	R			CRUD	R	R	R			R	R		R		R
Product Description	R			CRUD	RU	R	R			R	R		R		R
Product Unit of Measure	R			CRUD	R	R	R			R	R		R		R
Product Current Unit Price	R			CRUD	R		R			R	R		R		R
Product Quantity on Hand	X					RU	R			R	RU		R		RU

INDV=individual ALL=ALL SS=subset X=no access
S=submit C=create R=read U=update D=delete

图 7-24 数据-地点-CRUD 矩阵

在有些方法学和 CASE 工具中，可以为每个地点定义数据模型的视图。一个视图仅仅包含供单个地点访问的实体和属性。如果定义了视图，它们还必须与主数据模型保持同步（大多数 CASE 工具自动完成这一工作）。

复习题

1. 逻辑模型和物理模型之间有什么区别？
2. 为什么创建系统的一个实现无关的模型很重要？
3. 为什么需要创建系统的一个实现相关的模型？
4. 什么是实体？什么是实体实例？
5. 关系是实体之间的自然业务联系。学生和老师之间的关系是什么？这个关系依赖于学生能够上多少课、或者老师能够教多少课吗？
6. 什么是基数？举例说明。

问题和练习

1. 一个学生的姓的合理的数据属性域是什么？
2. 你会为一个学生的姓选择什么默认值？
3. 你为性别选择什么默认值？
4. 你正在处理的学生表包括以下属性：STUDENT ID、NAME、PHONE NUMBER 和 MAJOR。将它归一化为 3NF。
5. 在描述电影的表中你将使用什么属性？
6. 一个多对多关系（又称非特定关系）能够而且一般应该被消解成一对同一个关联实体的一对多关系。什么时候不成立？
7. 举一个多对多关系的例子。使用一个实体或者一个关联实体消解它。你使用哪种实体？为什么？
8. 描述第一范式到第三范式。分别举例说明。
9. 一位客户到一个鞋店买了几双鞋。绘制这个关系。
10. 请对三重关系、确定关系、非确定关系各举一个例子。
11. 从表面上看，数据建模并不需要多少创造性。为什么不正确？
12. 设计良好的数据库会给企业带来战略优势吗？如何带来？

项目和研究

1. 去学校图书馆，在流通柜台让图书管理员打印一份他们为你保存的信息。上面存储了什么类型的数据？有什么使你感到吃惊的吗，或者有什么看上去与借书无关的信息吗？如果有，请问为什么收集这些信息。
2. 从你在上一题中发现的信息中创建学生实体的属性列表。将它归一化成第三范式。
3. 到杂货店买一件东西。一个好的信息系统将对这个交易维护什么类型的数据？一个好的信息系统将为企业保存什么？
4. 将你在问题3中的答案至少同另一位学生的答案相比较。你们的答案有什么不同？
5. 哪些法律和隐私问题与杂货店使用的数据库相关？访问 <http://www.findlaw.com> 查询关于这个主题最近的一些法庭实例。将你的工作用一篇短文（5页）向你的同学介绍。
6. 数据库和数据库中保存的信息如何能被企业用于战略优势？在一个小组中，去一个你们选择的企业。讨论创建一个数据库，给公司提供一个现有问题或者探索企业的一个机会的解决方案（记住要有创造力）。

小型案例

1. 考虑一个虚构的称为 Wow Munchies 的在线杂货店。CIO 已经决定更新这个网上商店的数据库，以使它为不同部门收集相关的事务和销售信息。CIO 不确定什么软件或硬件能支持这个数据库。
第1步：你将如何决定在新数据库中收集和维持的哪些数据是重要的？要明确说明。
第2步：创建调查问卷等，把问卷交给受影响的部门的合适人选。检查每个部门使用的表格。
第3步：你得到了哪种回答？访问一个在线杂货店，看看“竞争对手”公司收集什么数据。你遗漏了什么数据吗？你得到的回答需要同部门人员进行第二次会议吗？如果需要，检查你的调查、问题和面谈。
第4步：利用你在上一问题中确定的方法，确定你需要包括在你的数据中的实体和属性。
第5步：绘制实体之间的关系和基数。你使用哪种数据模型？实现特定的，还是非特定的？为什么？
第6步：检查你的数据模型以使其没有多对多关系，并且模型归一化为第三范式。
- 第7步：向你的教授提交所有的问卷、调查、表格和回答，以及你最后的数据模型草图。包括一个简短的解释，关于你如何推导出你的实体、属性和关系，以及你的数据模型的任何假设或限制。
2. 研究一个汽车租赁代理，为它的汽车租赁数据库创建一个数据模型。汽车租赁会影响到什么部门？检查任何可公开获得的表格，创建一个备忘录，如果需要则进行面谈，以帮助你确定数据库应该包含什么内容。确保归一化成第三范式，消除了任何多对多关系。向教授提交你的数据模型和所有支持文档。
3. 考虑黑手党。假设那个有组织的犯罪团伙维护了一个数据库以逃避犯罪和运转他们的业务，他们应该保存什么信息？为什么？
4. 哪些法律的、道德的和隐私问题同犯罪斗争中使用的数据库相关？给你的班级研究并提交一份简短的论文（10页或少些）。

团队和个人练习

1. 在一个地理上和文化上分离的团队中进行项目管理是困难的。对于每个团队成员，分配成员到一个不同时区和不同语言的国家。假设所有的成员共用一种公共语言，尽管那种语言不是所有成员的母语。例如国家：美国、印度、以色列、中国、巴基斯坦、伊朗、法国、秘鲁和日本。
2. 个人练习：据说我们自己的创造力的边界就是我们自己和我们的经验。也就是说，使我们成为我们自己的同时也限制着我们。这对创造力意味着什么？你将如何成为有创造力的，并自由地思想？
3. 个人或团队练习：再工程一个通常的生活过程。如果“天空是限制”，你如何改变这个过程？标示出旧过程中的每一步，然后列出新过程中的每一步（例如，化妆、打扫房子、逛商店、上课）。把你的笔记拿到班上进行圆桌讨论。

过程建模

本章概述和学习目标

在本章中，你将学到如何绘制数据流图，这是一种流行的过程模型，它记录了系统的过程以及过程的数据流。本章将介绍以下内容：

- 定义过程建模并解释其优点。
- 认识并理解过程模型的基本概念和构造。
- 阅读并解释数据流图。
- 解释什么时候构造过程模型以及在哪里存储过程模型。
- 构造上下文图来说明系统与其环境的接口。
- 确定系统的用例、外部业务事件和临时业务事件。
- 实施事件划分，并将事件组织成功能分解图。
- 绘制事件图，然后将事件图合并成系统图。
- 绘制基本数据流图，并相应地按照数据结构和过程逻辑（结构化的英语和决策表）描述基本数据流和过程。

本章关键术语

逻辑模型（logical model）是描述系统是什么或者系统做什么的非技术性的图形化表示。同义词包括本质模型、概念模型和业务模型。

物理模型（physical model）是展示系统是什么或者系统做什么，以及系统如何实现的技术性的图形化表示。同义词包括实现模型和技术模型。

数据流图（Data Flow Diagram, DFD）是一种描述通过系统的数据流以及系统实施的工作或处理过程的过程模型。同义词包括泡式图、转换图和过程模型。

外部代理（external agent）是与系统交互的外部的人员、组织部门、其他系统或者其他组织，也称为外部实体。

数据存储（data store）存储数据供日后使用。同义词包括文件和数据库。

过程（process）是在输入数据流或条件上执行，或者对输入数据流或条件做出响应的工作，同义词是转换。

分解（decomposition）是将一个系统分解成子系统的行动。

分解图（decomposition diagram）是一种用来描述系统分解的工具，也称为层次图。

功能（function）是企业的一套相关的和正在进行的活动。

事件（event）是必须作为一个整体完成的逻辑单位工作，有时称为事务。

基本过程（elementary process）是为完成一个事件的响应所需要的离散的详细活动或任务。

数据流（data flow）是一个过程的数据输入，或者来自一个过程的数据（或信息）输出。

组合数据流（composite data flow）是由其他数据流构成的数据流。

控制流（control flow）表示触发一个过程的条件或非数据事件。

数据守恒（data conservation）是确保一个数据流只包含接收数据的过程真正需要的数据的实践。

数据属性（data attribute）是对最终用户和业务有意义的最小数据块。

数据结构（data structure）是数据属性的特定排列，它定义了数据流的一个实例。

分支的数据流（diverging data flow）是一个分成多个数据流的数据流。

合并的数据流 (converging data flow) 是多个数据流合并成一个数据流后的数据流。

事件划分 (event Partitioning) 是一种结构分析策略, 根据业务事件和对事件的响应将一个系统划分成子系统。

上下文数据流图 (context data flow diagram) 是用来记录系统的范围的过程模型, 也称为环境模型。

参与者 (actor) 是任何需要同系统交互的事物。

事件图 (event diagram) 是描述一个事件的上下文图的数据流图。

平衡 (balancing) 要求不同详细程度的数据流图保持一致性和完整性。

结构化英语 (Structured English) 是一种语言语法, 用于说明过程逻辑。

决策表 (decision table) 是一张表格, 说明了一组条件及其对应的行动。

8.1 过程建模简介

过程建模是一种组织和记录数据的结构和流向的技术, 它记录系统的“过程”和/或由系统的“过程”实现的逻辑、策略和程序。在信息系统构件上下文中 (见本章开始的“主页”), 逻辑过程模型用来记录从系统所有者和系统用户的观点看待的信息系统“过程”焦点 (“过程”列与系统所有者和系统用户行的交叉)。还请注意一类特殊的过程模型, 称为上下文图, 它说明了从系统所有者和系统用户的观点看待的“通信”焦点。在本章中, 我们将主要介绍系统分析的过程模型, 即数据流图。

数据流图是一种描述通过系统的数据流以及系统实施的工作或处理过程的工具。同义词包括泡式图、转换图和过程模型。我们还将介绍一个 DFD 规划工具, 称为分解图。最后, 我们将研究上下文图 (一个过程类模型), 它说明了系统与企业 and 外界 (包括其他信息系统) 的接口^①。

图 8-1 演示了一个简单的数据流图。在设计阶段, 这些业务过程中有一些可能会实现成计算机软件 (或者内部构造或者向软件供应商购买)。查看这张数据流图, 你会发现它容易阅读 (甚至在你学完本章之前) ——这一直是 DFD 的优点。图中仅有三种符号和一种连接^②:

- 圆角矩形表示要完成的过程或者工作。
- 正方形表示外部代理——系统的边界。
- 开放的方框表示数据存储——有时称为文件或者数据库。如果你已经阅读了第 8 章, 就应该发现这些数据存储对应了数据模型中的每个实体的所有实例。
- 箭头表示数据流——或者输入和输出, 到过程和来自过程。

不要混淆数据流图和流程图。程序设计常常使用流程图, 但数据流图完全不同。

数据流图已经流行了 20 多年了, 但是对 DFD 的兴趣最近又有所恢复, 这要归因于它们在业务过程重构 (business process redesign, BPR) 中的应用。当企业开始意识到大部分的数据处理系统仅仅是自动化了过时、低效而且官僚主义的业务过程时, 就会重新产生理顺那些业务过程的兴趣, 这首先是通过了达到分析、重构和/或改进业务过程的目的而建模那些业务过程来实现。结果, 信息技术可以以创造性的方式应用到改进的业务过程中, 这给企业带来极大的价值。在本章末尾, 我们将重新返回来讨论这个发展趋势。

8.2 过程建模的系统概念

8.2.1 外部代理

所有的信息系统都响应系统环境中的事件和条件。一个信息系统的环境包括外部代理, 它们形成了系统的边界, 并定义系统在哪里与其环境接口。**外部代理**定义位于项目范围之外但与正在被研究的系统交互的人、组织部门、其他系统或者其他组织。外部代理提供了进入一个系统的净输入, 并从一

① 在传统的结构化分析中, 上下文图被认为是另一类过程模型。但在面向对象分析中, 它们说明了范围和接口。在本书这一版里, 我们选择了后一种定义。

② DFD 有几种相互竞争的符号集, 本书采用了 Gane 和 Sarson (结构化分析) 的符号, 原因是它很流行。

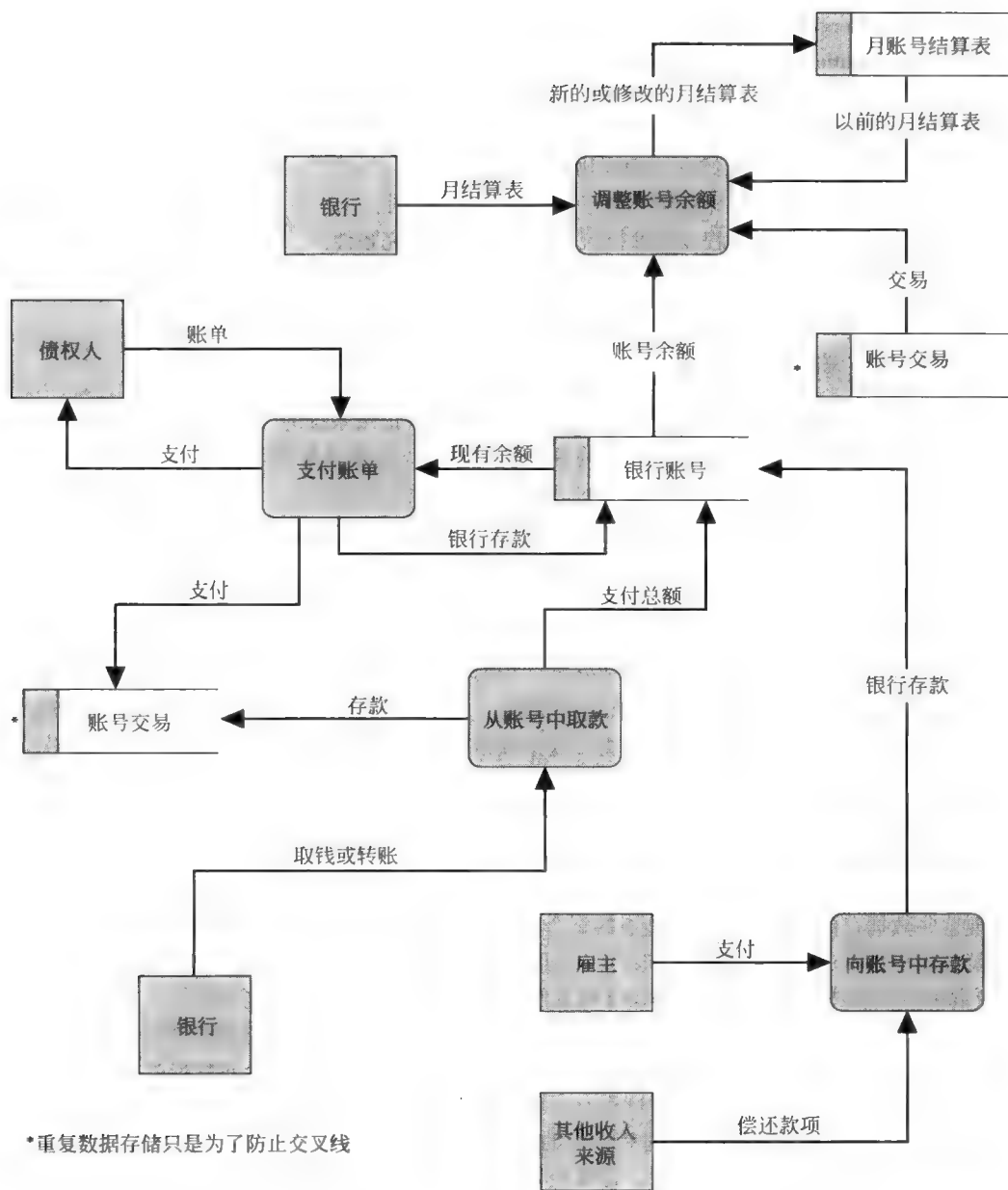


图 8-1 一个简单的数据流图

个系统接收净输出。常见的同义词包括外部实体（不要同第 7 章介绍的数据实体混淆）。

外部这个词意味着“在正被分析或设计的系统之外”。在实践中，外部代理可能实际位于企业外部（例如政府部门、客户、供应商和外部代理联系人），或者它可能在企业内部但在项目和系统范围之外（例如其他部门、其他业务功能和其他内部信息系统）。外部代理在数据流图 Gane 和 Sarson 符号中用正方形表示。DeMarco/Yourdon 的等价表示是矩形，如右图所示。

认识到工作和活动发生在外部代理内部是很重要的，但那些工作和活动称为是“范围之外的”而且不会变化。因此，系统和这些边界之间的数据流不应



该引起由外部代理执行的工作和活动发生实质性改变。

在一个逻辑数据流图中的外部代理可能包括系统必须交互的人、业务部门、其他内部系统以及外部组织，它们包括在逻辑 DFD 中意味着系统要与这些代理交互。外部代理一般是下列内容之一：

- 一个办公室、部门、分部或个人，它们在公司内给系统提供净输入，从系统接收净输出，或者二者兼备。
- 位于你的公司以外但给你的系统提供净输入或者从你的系统接收净输出的组织、机构或者个人，例如客户、供应商、承包人、银行和政府部门。
- 另一个企业或者信息系统——可能（尽管不是必需的）是基于计算机的系统——那个系统独立于你的系统，但你的系统必须与它交互。很少有信息系统不与其他信息系统接口，与其他企业的信息系统接口也越来越常见。
- 系统的最终用户或管理人员之一。在这种情况下，用户或管理人员是要输入到系统的净数据源和/或由系统产生的净输出的目的地。

外部代理应该用描述性的单数名词描述，例如：注册员、供应商、生产系统或财务信息系统。外部代理表示了固定的物理系统，所以它们可以采用十分实际的名字或缩写——甚至在“逻辑”DFD 中！例如，一个代表我们学校的财务管理信息系统的外部代理将被称为“FMIS”。如果外部代理描述的是一个人，我们推荐使用工作职务或者角色名称，而不用真实名字（例如，使用“会计员”，而不是“Mary Jacobs”）。

为了避免 DFD 中的数据流线交叉，允许在 DFD 上重复外部代理。但作为一条通用规则，外部代理应该放置在页面的周围，与它的系统边界的定义相一致。

8.2.2 数据存储

大多数信息系统收集数据供日后使用。数据保存在数据存储中，数据存储是数据流图中的最后一个符号。它使用开口的方框表示，如右图所示。数据存储是一个数据的“仓库”。同义词包括文件和数据库（尽管那些词对于基本过程建模来说太面向实现了）。如果数据流是运动中的数据，那么数据存储就是静止的数据。

理想情况下，基本的数据存储应该描述关于企业想存储数据的“事物”。这些事物包括：

人：“代理”、“承包人”、“客户”、“部门”、“分部”、“雇员”、“导师”、“办公室”、“学生”、“供应商”。注意，人实体可以表示个人、小组或组织。

地点：“销售地区”、“建筑物”、“房间”、“分支办公室”、“校园”。

对象：“图书”、“机器”、“部件”、“产品”、“原材料”、“软件许可证”、“软件包”、“工具”、“汽车模型”、“汽车”。对象实体可以表示实际的对象（例如“软件许可证”），或者一类对象的说明（例如“软件包”）。

事件：“应用”、“奖励”、“取消”、“分类”、“飞行”、“发票”、“订单”、“注册”、“续借”、“获取”、“预订”、“销售”、“旅行”。

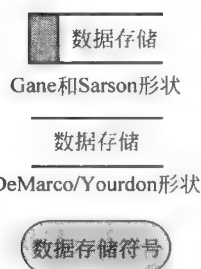
概念：“账号”、“时间段”、“债券”、“课程”、“基金”、“资格”、“股票”。

注意 如果上面的清单看上去有点面熟，确实是！数据存储表示了数据实体的所有具体值——在第7章中定义为我们想存储数据的事物。因此，数据存储同步了系统的过程模型和数据模型。

如果在过程建模之前进行数据建模，那么大部分数据存储的确定就可以通过如下规则进行简化：

实体关系图中的每个实体都应该有一个数据存储（在我们的模型中，甚至包括了关联实体和弱实体数据存储）。

如果在数据建模之前进行过程建模，那么获取数据存储就往往缺乏根据。在这种情况下，我们最



好的建议是确定文件或数据存储（例如，计算机文件和数据库、文件柜、书、目录等）的现有实现，然后重命名它们以反映有关存储数据的逻辑“事物”。同信息工程策略相一致，我们推荐在过程模型之前建立数据模型。

通常，数据存储按照相应的数据模型实体的复数形式命名。因此，如果数据模型包含一个实体“客户”，过程模型将包括一个数据存储“一些客户”。这是合理的，因为按照定义，数据存储存储了实体的所有实例。避免使用物理性词汇，例如文件、数据库、文件柜、文件夹等。

同外部代理的情况一样，在 DFD 图中允许复制数据存储以避免数据流线的交叉，但应该尽量减少重复。

8.2.3 过程概念

回顾第1章内容可知，“过程”是一个基本的信息系统构件。所有的信息系统都包含过程——通常有很多过程。信息系统过程响应业务事件和条件，并将“数据”（另一个构件）转换成有用的信息。我们需要一种方法来建模过程并理解过程与系统环境、其他系统以及其他过程如何交互。



8.2.3.1 系统是过程

本书使用“系统”这个词描述几乎任何有序组织的想法或结构。人们常常提到教育系统、计算机系统、管理系统、商业系统，当然还有信息系统。在系统模型最早最简单的形式中，系统就是过程。

在系统分析中，模型用来显示或者表现一个系统。如图8-2所示，一个系统的最简单的过程模型是基于输入、输出和系统本身的——被看作是一个过程。过程符号定义了系统的边界，系统在这个边界之内，环境在边界之外，系统与其环境交换输入和输出。因为环境总是在变化，设计良好的系统具有一个反馈和控制环路，以使系统可以自我调整适应变化的情况。

例如，把一个企业考虑为一个系统，该系统在一个包含了客户、供应商、竞争者、其他行业和政府的环境中运行，它的输入包括材料、服务、新雇员、新设备、工具、费用和订单（仅列出了一些输入），输出包括产品和/或服务、废料、淘汰设备、以前的雇员和费用（支付）。企业监视环境以对其产品线、服务、操作程序、竞争者、经济做出相应调整。

在本章中，我们使用圆角矩形（Gane 和 Sarson 符号）表示过程，如上图所示。过程是在输入数据流或条件上执行，或者对输入数据流或条件做出响应的工作，同义词是转换。其他的过程建模符号使用圆（DeMarco/Yourdon 符号）或者矩形（SSADM/IDEFO 符号）表示过程。符号的选择往往取决于所使用的方法和 CASE 工具的特征。

尽管过程可以由人、部门、机器人、机器或者计算机实施，但我们再一次希望关注正实施什么工作或行动（逻辑过程），而不是谁或什么正在做工作或活动（物理过程）。例如，在图8-1中，我们包含了逻辑过程“从账号中取款”，但我们没有指出如何完成这个过程。我们可以考虑几种直观的物理实现，例如：使用 ATM 机、银行上门服务或者到银行去办理。

8.2.3.2 过程分解

当把一个复杂系统作为一个整体（即作为单个过程）看待时，通常很难全面地理解它。所以，在系统分析中，我们将一个系统分解成它的组件子系统，子系统又被分解成更小的子系统，直到确定出整个系统的可管理子集（见图8-3）。我们称这项技术为分解。分解是将一个系统分解成它的组件子系统、过程和子过程的行动。每个层次的抽象都揭示了有关整个系统或系统的某个子集的或多或少的细

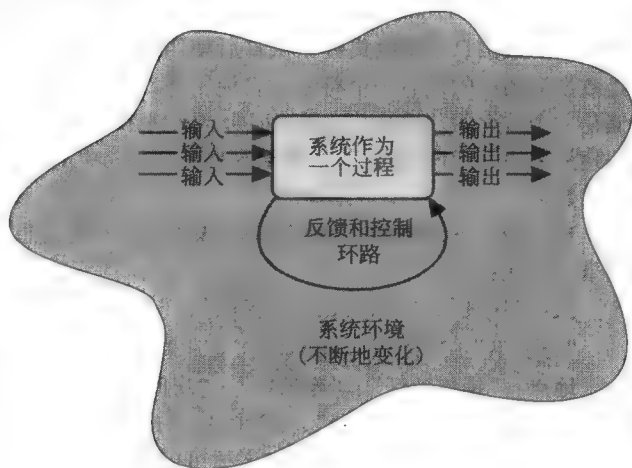


图8-2 一个系统的传统的过程模型

节（期望的细节）。你已经以各种方式应用过分解技术了。其中大多数人一定写过论文的提纲——这就是分解的一种形式。许多人也曾经把一个大中型规模的计算机程序分解成可以在集成之前独立开发和测试的子程序，这也是一种分解。

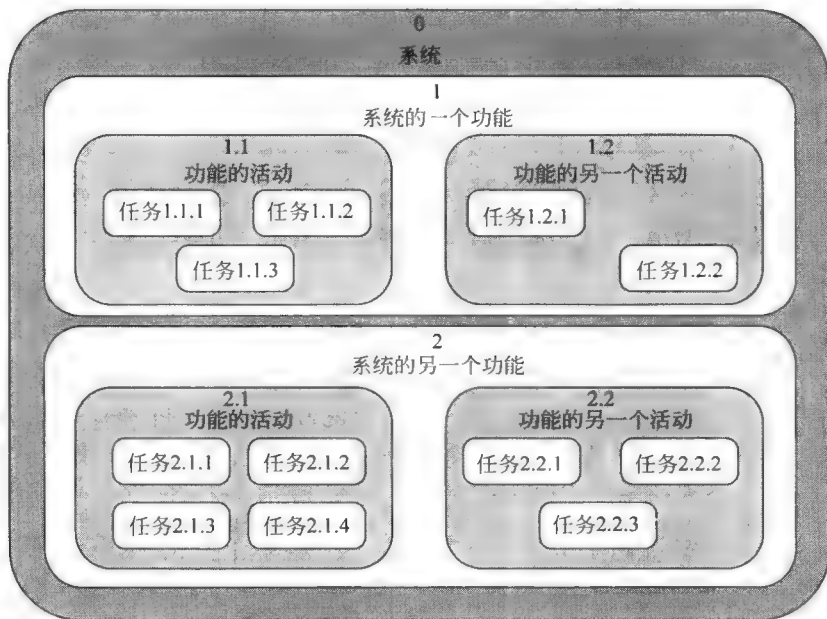


图 8-3 一个系统包括许多子系统和过程

在系统分析中，分解可以把一个系统划分成为用来改进沟通、分析和设计的逻辑子系统。但除了系统中的最小部分以外，构造类似图 8-3 的图形可能有点困难。图 8-4 演示了另一种布局，这种布局被许多 CASE 工具和开发方法学使用，被称为分解图。我们将在本章大量地使用这种图。分解图也称为层次图，显示了一个系统的自顶向下的功能分解和结构。分解图本质上是一种规划工具，用于更详细的过程模型（即数据流图）。以下规则可应用于分解图：

- 分解图中每个过程或者是父过程，或者是（父过程的）子过程，或者二者都是。
- 父过程必须有两个或多个子过程——单个子过程没有意义，因为这不能揭示系统的任何额外细节。
- 在大多数分解图标准中，一个子过程只可以有一个父过程。
- 最后，一个父过程的一个子过程可以是它自己的子过程的父过程。

图 8-4 中分解图上半部和下半部演示了两种过程和连线的布局风格。你可以按照表示一个整洁的模型的需要使用任一种风格，或者这两种风格都使用。为了尽可能清晰，有些模型可能需要使用多页纸来描述。

分解图中的连线不包含箭头，因为图用来显示结构，而非流程。而且，连线没有命名。隐含地它们都具有同样的名字——“由……构成”——由于一个父过程的子过程的总和等于父过程。

8.2.3.3 逻辑过程和命名规则

逻辑过程是无论你怎么实现这个系统都必须实施的工作或行动。逻辑过程的命名规则取决于过程在分解图/数据流图中的位置，以及描述的过程类型。存在三类逻辑过程：功能过程、事件过程和基本过程。

功能是企业的一套相关的和正在进行的的活动。功能没有开始和结束，它只是不断地按需要执行其工作。例如，一个生产系统可能包括以下功能（子系统）：生产计划、生产调度、材料管理、生产控制、质量管理和库存控制，而每一个功能都可能包括几十个或几百个更离散的支持特定活动和任务的过程。功能用描述整个功能的名词命名。其他例子如：订单输入、订单管理、销售报告、客户关系和退货退款。

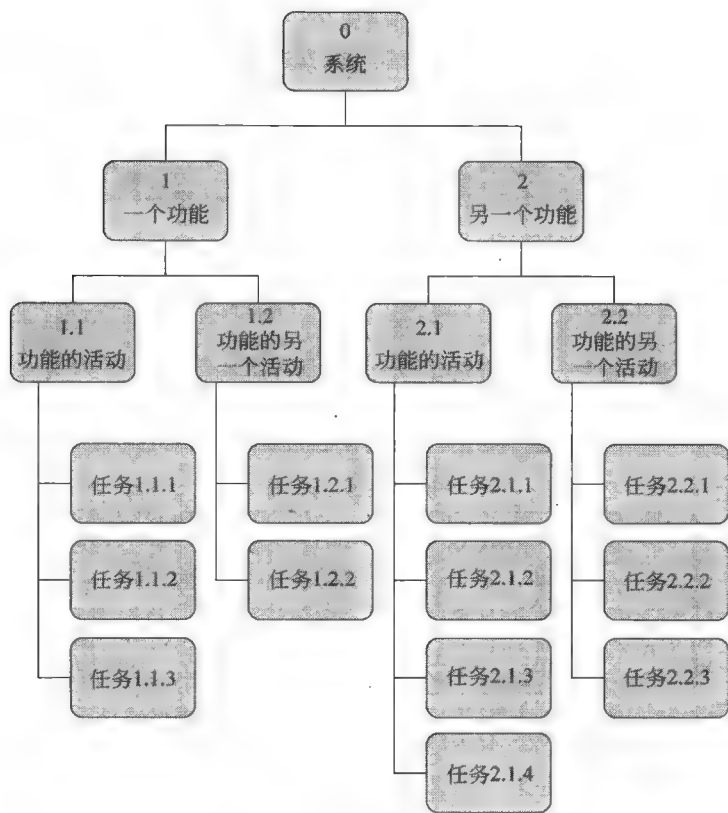


图 8-4 一个分解图（对应图 8-3）

事件是必须作为一个整体完成的逻辑单位工作。事件由离散的输入触发，当过程与相应的输出响应时事件结束。事件有时称为事务。功能由响应事件的过程组成。例如，“材料管理”功能可以响应下列事件：测试材料质量、库存新材料、处理毁坏的材料、处理损坏的材料、为生产申请材料、从生产返回未用的材料、订购新材料等。其中，每一个事件都有一个触发器和响应，它们可以用事件的输入和输出定义。

使用现代结构化分析技术，例如由 McMenamin、Palmer、Yourdon 和 Robertsons 推荐的技术，系统功能最终将被分解成业务事件。每个业务事件由响应那个事件的一个过程表示。事件过程名字的格式往往十分通用。我们将采用如下的规则命名事件过程：“处理 ____”、“响应 ____”或“生成 ____”，其中空格将填上事件（或事件相应的输入）的名字。事件过程名字的例子有：处理客户订单、处理客户订单修改、处理客户地址修改、响应客户投诉、响应订单查询、响应产品价格查询、生成退单报告、生成客户账号结算表和生成发票。

事件过程可以进一步分解成详细地说明系统必须如何响应事件的基本过程。**基本过程**是为完成一个事件的响应所需要的离散的详细活动或任务。换句话说，它们是在一个过程模型中描述的最低层次细节。常用的同义词是原子过程。基本过程应该用一个强动作动词后跟一个描述实施的工作的宾语从句命名。基本过程名字的例子有：验证客户身份、验证订购产品编号、检查产品可得到性、计算订单费用、检查客户信用、排序退单、获得客户地址、修改客户地址、增加新客户和删除客户。

逻辑过程模型忽略了那些除了移动或路由数据（也就是保持数据不变）而不做任何其他工作的过程。因此，应该忽略任何类似秘书或者办事员接收并简单地传递一些文档到下一个处理地点之类的过程。最后，应该仅仅保留以下逻辑过程：

- 执行计算（计算平均积分点）。
- 做出决策（决定是否可以获得订购的产品）。

- 排序、过滤或者总结数据（确定过期的发票）。
- 组织数据成为有用的信息（生成报告或者回答问题）。
- 触发其他过程（打开抽屉或者指挥一个机器人）。
- 使用存储的数据（创建、读取、修改或者删除记录）。

注意避免过程中常见的以下3种错误（如图8-5所示）。

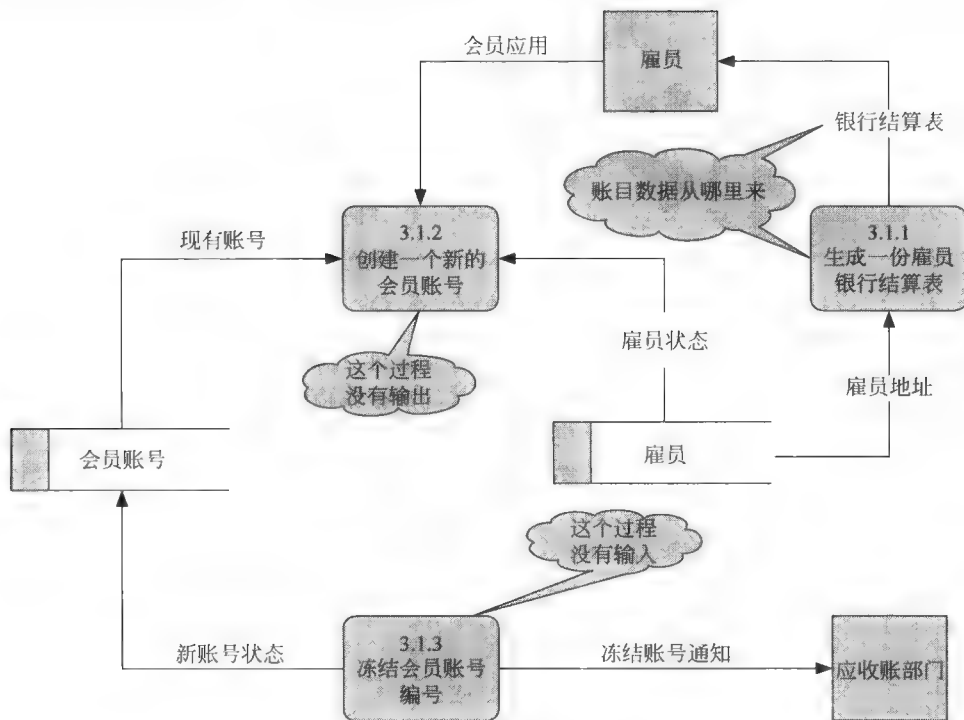


图8-5 数据流图中的常见错误

- 过程3.1.2有输入但没有输出。我们称之为黑洞，因为数据输入到过程，然后就消失了。在大多数情况下，建模人员只是忘了输出。
- 过程3.1.3有输出但没有输入。在这种情况下，输入流似乎被忘记了。
- 在过程3.1.1中，输入不足以产生输出。我们称之为灰洞，这有几种可能的原因，如：1）一个错误命名的过程；2）错误命名的输入和/或输出；3）不完全的事实。灰洞是最常见的错误——也是最使人难为的错误。一旦数据流图交给了程序员，到一个过程（被实现成一个程序）的输入数据流必须足以产生输出数据流。

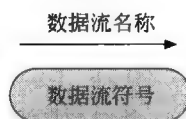
8.2.4 数据流

过程响应输入并产生输出。因此，所有的过程至少都有一个输入数据流和一个输出数据流。数据流是过程与系统环境之间的通信。下面介绍数据流的一些基本概念和规则。

8.2.4.1 运动中的数据

数据流是运动中的数据。系统与其环境之间或者系统内两个过程之间的数据流动是一种通信，下面研究这种形式的通信。

数据流表示到一个过程的数据输入，或者来自一个过程的数据（或信息）输出。数据流也用于表示在文件或数据库（在DFD中称为数据存储）中创建、读取、删除或修改数据。可以把数据流看作是一条“高速公路”，已知成分的报文在其上旅行，数据流名称还隐含了哪些类型的数据可以沿着“高速公路”旅行。这条“高速公路”则被描绘成一条带箭头的实线。



这个定义中的关键概念是报文。一起旅行的数据应该被表示为一个单一的数据流，无论其中可能包含多少物理文档。报文概念如图 8-6 所示，其中显示了一个逻辑数据流报文的正确表示方式和不正确表示方式。

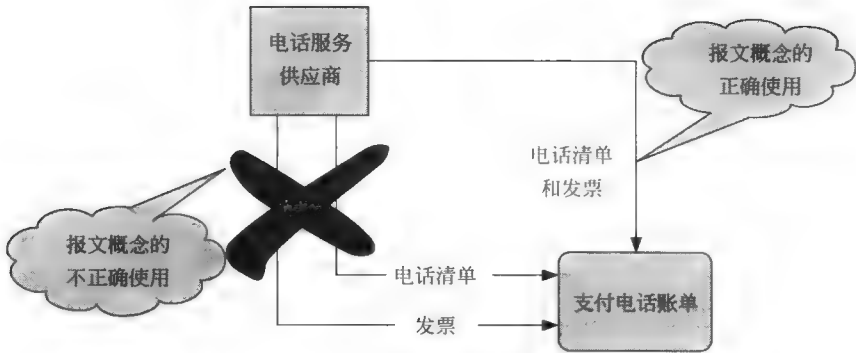


图 8-6 数据流报文概念

已知成分的概念也很重要。数据流或者由具体的数据属性（也称为数据结构——后面将进一步介绍）构成，或者由其他数据流构成。组合数据流是由其他数据流构成的数据流。它们用于在高层的数据流图中组合相似的数据流，以使数据流图更便于阅读。例如，在图 8-7a 中，一个高层的 DFD 将所有类型的订单合并为一个称为“订单”的组合数据流；在图 8-7b 中，一张更详细的数据流图显示了特定类型的订单：“长期订单”、“急需订单”和“标准订单”。这些不同的订单要求某些不同的处理（小的黑圆圈称为交合点，它指示任何给定的“订单”仅仅是某种订单的一个实例）。

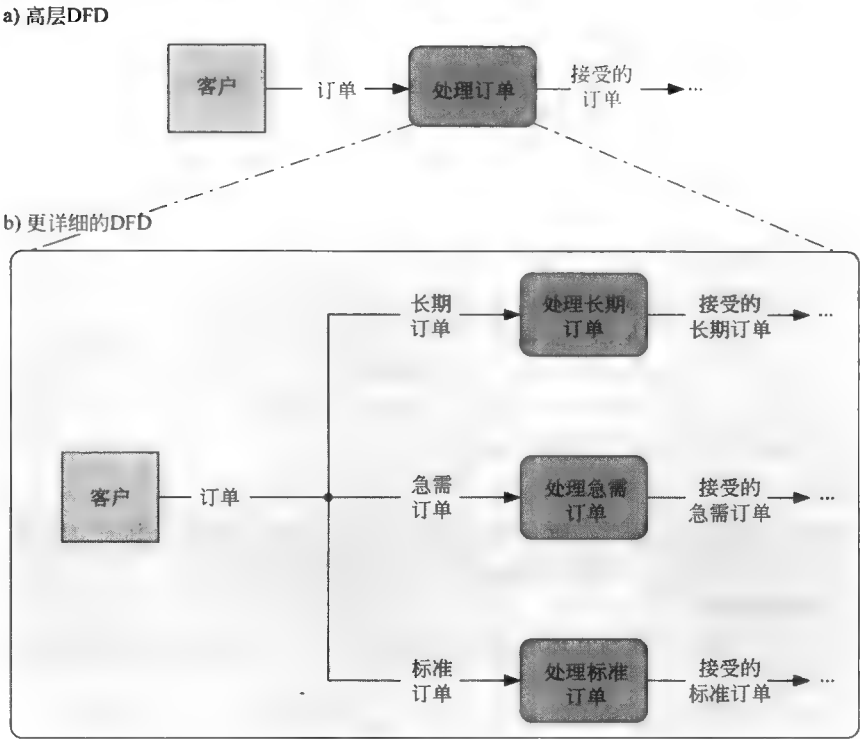
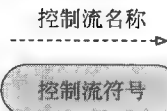


图 8-7 组合数据流和基本数据流

组合数据流的另一种常见用法是合并所有的报告和查询响应到一个或两个组合流。合并的原因有两个：第一，输出可能很多；第二，许多现代的系统提供了大量用户定义的报告和查询，这些报告和

查询在系统实现和使用之前无法预测。

某些数据流图方法也可以识别被称为控制流的非数据流。控制流表示触发一个过程的条件或非数据事件。可以把它看作是系统工作时的一个监控条件，当系统意识到那个条件满足了某些预定义的状态时，条件输入到的过程就会被启动。典型的例子是时间，例如：报告生成过程可以由时序事件“月末”触发。在实时系统中，控制流经常表示实时条件，例如“温度”和“高度”。在大多数区分数据流和控制流的方法学中，控制流用一条带箭头的虚线表示。



通常，信息系统分析员主要处理数据流；但是，随着信息系统和实时系统（例如生产过程和计算机集成制造）更多地集成在一起，区分控制流也就变得必需了。

8.2.4.2 逻辑数据流和规则

虽然我们认识到数据流可以用多种方式（例如，电话呼叫、业务表格、条形码、备忘录、报告、计算机屏幕以及计算机到计算机通信）实现，但是我们只对逻辑数据流感兴趣。因此，我们只对需要的流感兴趣（而非我们将如何实现这个流），所以不应该鼓励数据流名称过早地涉及任何可能的实现。

数据流名称应该采用描述性的单数名词和名词短语，不要采用复数名词。我们不想暗示流的出现必须实现成物理的一批。

数据流名称也应该是唯一的。形容词和副词的使用有助于描述处理如何改变一个数据流。例如：如果到一个过程的输入命名为“订单”，输出就不应该再命名为“订单”，它可以命名为“有效的订单”、“认可的订单”、“具有有效产品的订单”、“获得信用认可的订单”或者任何其他反映过程对原始订单处理的描述性名词。

往来于数据存储的逻辑数据流需要特殊的命名考虑，如图8-8所示（编号的圆圈对应如下的注解）。

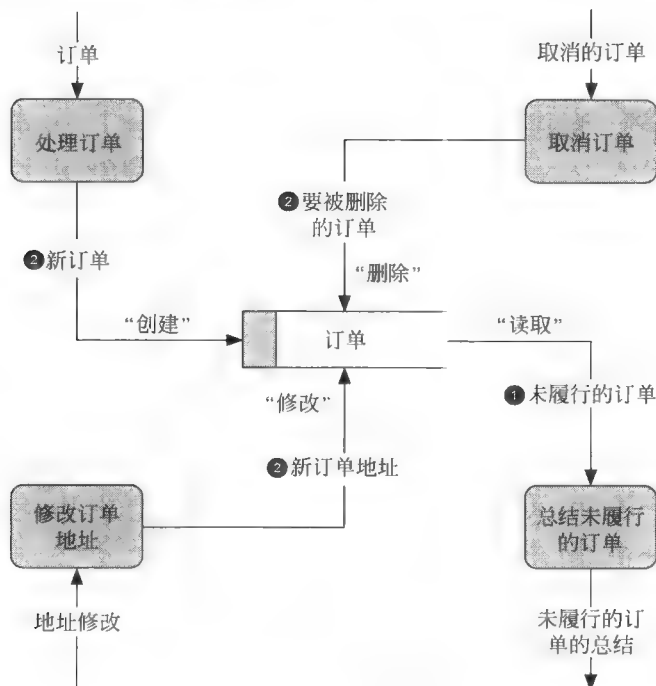


图 8-8 往返于数据存储的数据流

- 只显示了净数据流。你可能意识到需要获得一个记录以便修改或删除它，但是除非数据需要用于某些其他的目的（例如计算或决策），否则“读取”动作就不必显示出来。这样做使得图形整洁。
- ① 从一个数据存储到一个过程的数据流指示了那个数据被“读取”以用于某些特殊的用途，数据流名称应该清楚地指出读取了什么数据。这在图8-8中显示。

② 从一个过程到一个数据存储的数据流指示了那个数据在/从那个数据存储中被创建、删除或修改。如图 8-8 所示，应该清楚地命名这些数据流以反映执行的特定动作（例如“新客户”、“要被删除的客户”或“修改的订单地址”）。

注意那些名称暗示了在一个文件上可以实施的典型动作，也就是：“创建”、“读取”、“修改”、“删除”（CRUD）。在一个实际的 DFD 中，我们不会在图中记录这些动作名称。

所有的数据流都应该命名，未命名的数据流通常是流程图思想的产物（例如，第一步、第二步等）。如果你不能给数据流起一个合理的名字，它可能就不存在。同逻辑建模的目标相一致，数据流名称应该描述数据流，而不是描述数据流如何或者可以实现。例如，假设用户这样解释他们的系统：“我们填写表格 23 三遍，并把它发送出去……”“表格 23”这个数据流的逻辑名称可能是“课程请求”。这个逻辑名称消除了物理的实现偏见——我们必须使用一个纸张表格，或者使用复写纸拷贝的提示。最终，这将使我们能自由地选择其他物理实现，例如按键式电话响应、联机注册屏幕甚至电子商务因特网网页。

最后，所有的数据流必须以一个过程开始和/或结束，因为数据流是过程的输入和输出。因此，图 8-9 左边的所有的数据流都是非法的，改正错误后的图在右边显示。

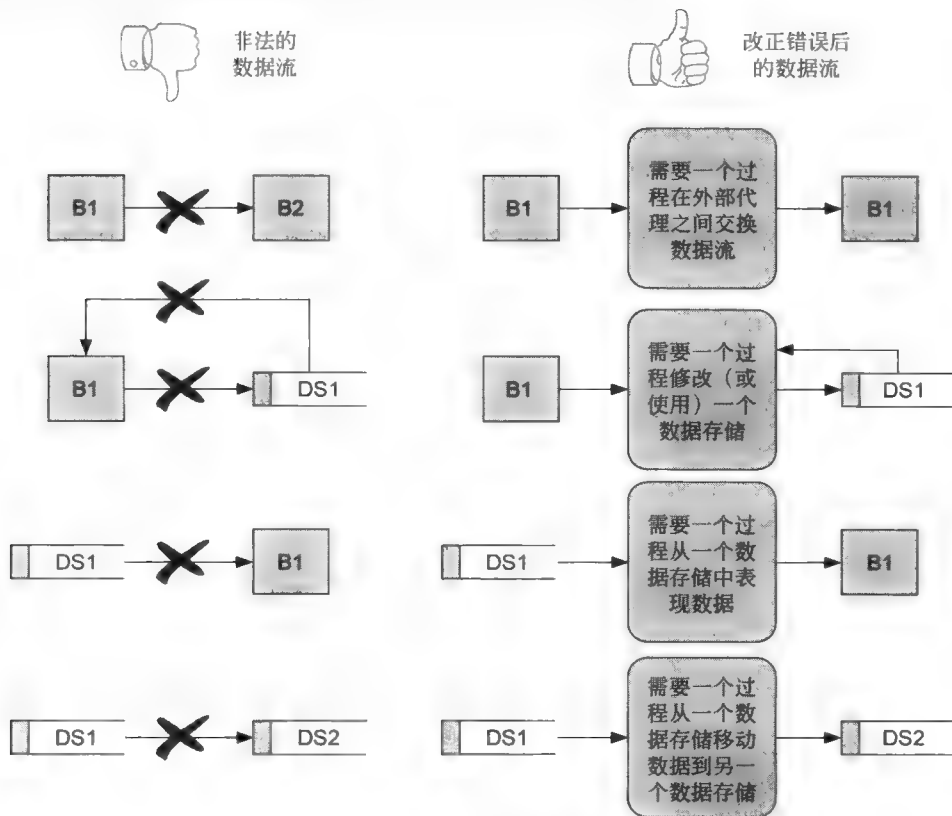


图 8-9 非法数据流

8.2.4.3 数据流的守恒

多年来，我们一直试图通过自动化改进业务过程。但自动化并不总是可行或者工作得很好，因为业务过程是在计算机出现以前被设计用来处理数据流的。例如考虑普通的业务表格。表格分成为不同用户设计的小节很常见。第一位收到表格的人完成表格中他所要填写的部分，下一位收到表格的人完成他那部分，等等。在这个处理序列的某一点上，表格甚至会被撕开并送给另一个接收者，这个接收者填写一个新的多部分表格，而这个新表格又需要从初始表格转录许多同样的数据。在大学中，我们已经看到过这样的例子：拙劣的表格设计要求相同的数据输入多遍。

现在，如果数据流是根据目前的业务表格和过程计算机化的，则得到的计算机程序将仅仅是自动

化了这些低效率。这正是在大多数企业中发生的事情！如今，重新对业务过程重构的强调鼓励管理人员、用户和系统分析员在设计任何信息系统之前确定并消除这些低效率因素。我们可以通过在逻辑数据流图中实践数据守恒支持这种发展趋势。数据守恒，有时称为“饥饿过程”，要求一个数据流只包含接收数据的过程真正需要的数据。通过保证过程只收到它们真正需要的那么多数据，我们简化了过程之间的接口。为了实践数据守恒，我们必须精确地定义每个（非组合）数据流的数据成分。数据成分以数据结构的形式表示。

8.2.4.4 数据结构

最终，一个数据流包含称为属性的数据项。数据属性是对最终用户和业务有意义的最小数据块（这个定义也可应用于第7章中介绍的属性）。数据流“订单”的例子属性可以包括“订单编号”、“订单日期”、“客户编号”、“发货地址”（它又包括属性，例如“街道地址”、“城市”和“邮政编码”）、“订购的产品编号”、“订购数量”等等。注意有些属性对“订单”的每个实例出现一次，而有些属性可能对“订单”的单个实例出现多次。

构成一个数据流的数据属性被组织成数据结构。数据流可以按照下列类型的数据结构描述：

- 一个序列或者一组依次出现的数据属性。
- 从一组属性中选择一个或多个属性。
- 一个或多个属性的重复。

最常用的数据结构符号是许多 CASE 工具使用的布尔代数符号。有些 CASE 工具和方法学支持特殊的符号记法，但本质上仍是等价的符号。图 8-10 展示了数据流“订单”的一个数据结构例子，其中使用了如下符号：

- = 意思是“包含”或者“由……构成”。
- + 意思是“并且”，指明了序列结构。

数据 结 构	解 释
ORDER = ORDER NUMBER + ORDER DATE + [PERSONAL CUSTOMER NUMBER, CORPORATE ACCOUNT NUMBER] SHIPPING ADDRESS = ADDRESS + (BILLING ADDRESS = ADDRESS) + 1 PRODUCT NUMBER + PRODUCT DESCRIPTION + QUANTITY ORDERED + PRODUCT PRICE + PRODUCT PRICE SOURCE + EXTENDED PRICE N + SUM OF EXTENDED PRICES + PREPAID AMOUNT (CREDIT CARD NUMBER + EXPIRATION DATE) (QUOTE NUMBER) ADDRESS = (POST OFFICE BOX NUMBER) + STREET ADDRESS + CITY + [STATE MUNICIPALITY] + (COUNTRY) + POSTAL CODE	ORDER 的一个例子包括： ORDER NUMBER 和 ORDER DATE 和 或者 PERSONAL CUSTOMER NUMBER 或 CORPORATE ACCOUNT NUMBER 和 SHIPPING ADDRESS（等于 ADDRESS） 和（可选的）：BILLING ADDRESS（等于 ADDRESS） 和 一个或若干个实例： PRODUCT NUMBER 和 PRODUCT DESCRIPTION 和 QUANTITY ORDERED 和 PRODUCT PRICE 和 PRODUCT PRICE SOURCE 和 EXTENDED PRICE 和 SUM OF EXTENDED PRICES 和 PREPAID AMOUNT 和 （可选的）：CREDIT CARD NUMBER 和 EXPIRATION DATE 和 （可选的）：QUOTE NUMBER ADDRESS 的一个例子包括： （可选的）：POST OFFICE BOX NUMBER 和 STREET ADDRESS 和 CITY 和 STATE 或 MUNICIPALITY 和（可选的）：COUNTRY 和 POSTAL CODE

图 8-10 一个数据流的数据结构

- [...] 意思是“方括号内的属性只有一个可以出现”——指明了选择结构。方括号中的属性用逗号分隔。
- {...} 意思是括号中的属性对于这个数据流的一个实例可以出现多次——指明了重复结构。括号中的属性用逗号分隔。
- (...) 意思是圆括号中的属性对这个数据流的某些实例来说是可选的——没有值。

根据我们的经验，所有的数据流都可以用这些基本结构描述。图 8-11 使用例子演示了每个基本结构。回到图 8-10，注意结构被组合起来描述这个数据流的数据内容。

数据结构	例 子 (相关部分用 黑体 表示)	解 释 (相关部分用 黑体 表示)
属性序列 ——序列数据结构指出可能 (或者必须) 包括在一个数据流中的一个或多个属性	WAGE AND TAX STATEMENT = TAXPAYER IDENTIFICATION NUMBER + TAXPAYER NAME + TAXPAYER ADDRESS + WAGES, TIPS, AND COMPENSATION + FEDERAL TAX WITHHELD + ...	WAGE AND TAX STATEMENT 的一个例子包括: TAXPAYER IDENTIFICATION NUMBER 和 TAXPAYER NAME 和 TAXPAYER ADDRESS 和 WAGES, TIPS, AND COMPENSATION 和 FEDERAL TAX WITHHELD 和...
属性选择 ——选择数据结构允许显示不同的属性集描述数据流的不同实例	ORDER = (PERSONAL CUSTOMER NUMBER, CORPORATE ACCOUNT NUMBER) + ORDER DATE + ...	ORDER 的一个例子包括: 或者 PERSONAL CUSTOMER NUMBER 或者 CORPORATE ACCOUNT NUMBER; 和 ORDER DATE 和...
属性重复 ——重复数据结构用来设定在数据流的一个实例中可以 (或者必须) 重复多次的数据属性或者属性组 重复的最小次数通常是 0 或 1 重复的最大次数可以用 “n” 表示, 意思是 “多个”, 其中实例的实际数量对每个数据流实例不同	CLAIM = POLICY NUMBER + POLICYHOLDER NAME + POLICYHOLDER ADDRESS + 0 DEPENDENT NAME + DEPENDENT'S RELATIONSHIP N + 1 EXPENSE DESCRIPTION + SERVICE PROVIDER + EXPENSE AMOUNT N	CLAIM 的一个例子包括: POLICY NUMBER 和 POLICYHOLDER NAME 和 POLICYHOLDER ADDRESS 和 零个或者若干个实例: DEPENDENT NAME 和 DEPENDENT'S RELATIONSHIP 和 一个或者若干个实例: EXPENSE DESCRIPTION 和 SERVICE PROVIDER 和 EXPENSE ACCOUNT
可选属性 ——可选符号指出在一个序列或选择数据结构中的一个数据属性或一组数据属性可以不被包含在一个数据流的所有实例中 注意: 对于重复数据结构, 最小值 “0” 与整个重复组 “可选” 一样	CLAIM = POLICY NUMBER + POLICYHOLDER NAME + POLICYHOLDER ADDRESS + (SPOUSE NAME + DATE OF BIRTH) + ...	CLAIM 的一个例子包括: POLICY NUMBER 和 POLICYHOLDER NAME 和 POLICYHOLDER ADDRESS 和 (可选的) SPOUSE NAME 和 DATE OF BIRTH 和...
可复用属性 ——对于包含在许多数据流中的属性组, 希望创建一个可以在其他数据结构中复用的独立数据结构	DATE = MONTH + DAY + YEAR	可复用结构可包括在其他数据流结构中, 如下所示: ORDER = ORDER NUMBER... + DATE INVOICE = INVOICE NUMBER... + DATE PAYMENT = CUSTOMER NUMBER... + DATE

图 8-11 基本数据结构

为每个数据流定义数据结构是很重要的——你正在为每个输入和输出定义业务数据需求！这些需求必须在任何过程可以实现成计算机程序之前就确定下来。这个标准的记法为用户和程序员的沟通提供了一种简单而有效的方法。

8.2.4.5 域

属性就是一块数据。当分析系统时，我们应该为属性定义合法的或者有意义的值。每个属性的值用两个特性来定义：数据类型和域。属性的数据类型定义了什么类型的数据可以存储在那个属性中。属性的域定义了属性可以合法地取什么值。数据类型和域的概念在 7.2.2.1 节介绍过，请参考第 7 章内容以及表 7-1 和表 7-2 中有关数据类型和域的更为完整的描述。

8.2.4.6 分支流和合并流

在数据流图中，有时需要描述分支的数据流或合并的数据流。分支的数据流是一个分成多个数据流的数据流。分支的数据流指示了一个数据流的所有或者部分路由到不同目的地^①。合并的数据流是多个数据流合并成一个数据流后的数据流。合并的数据流指示了不同来源的数据流可以（必须）合并成一个报文供后续处理。

图 8-12 显示了分支的数据流和合并的数据流。注意我们没有包含一个过程来“路由”这些数据流，数据流仅仅简单地从一个公共的数据流分开，或者合并到一个公共的数据流。本书中使用了下面的符号（它们没有被所有的 CASE 工具支持）：

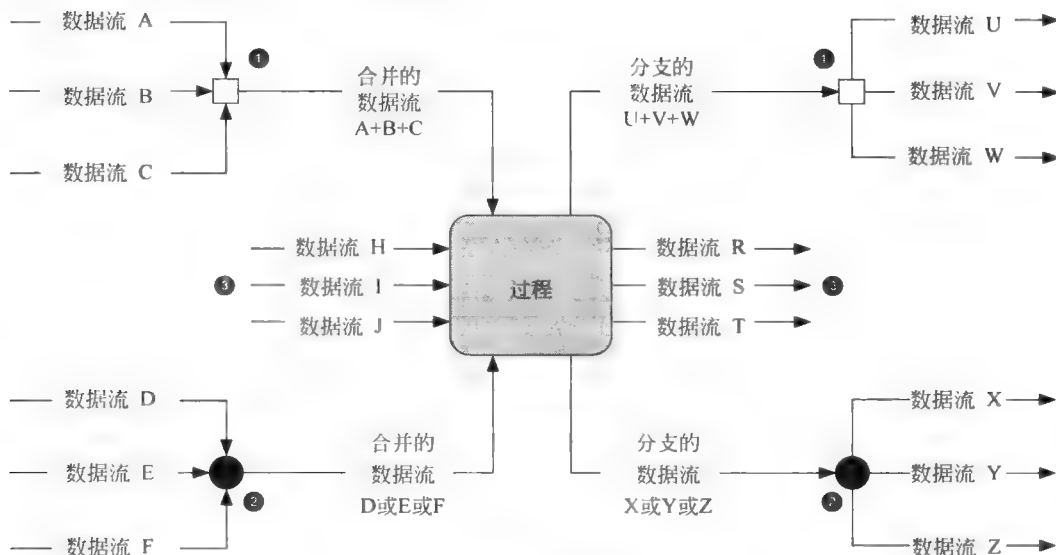


图 8-12 分支的数据流和合并的数据流

- ① 小方块交叉点意思是“与”。这意味着每次执行过程时，必须输入（或输出）所有的分支数据流或合并数据流（有些 DFD 记法在数据流之间放置一个“+”号）。
- ② 小黑色交叉点意思是“互斥或”。这意味着每次执行过程时，必须只输入（或输出）分支数据流或合并数据流中的一个（有些 DFD 记法在数据流之间放置一个“*”号）。
- ③ 在没有分支数据流或合并数据流的情况下，读者应该假设存在一个“包含或”。这意味着每次执行过程时，可以输入任何或者所有的数据流。

使用上面的规则，再复杂的业务过程和数据流组合都可以被画出来。

① 有些专家建议只有在数据流中的所有数据都被路由到所有的目的地时才使用分支的数据流。我们倾向于经典的 DeMarco 定义，其中允许数据流的所有或者部分路由到不同的目的地。

8.3 逻辑过程建模的过程

在系统分析部分和本章中，我们主要专注于作为业务需求分析一部分的逻辑过程建模。在信息系统框架中，逻辑过程模型具有一个“过程”焦点和“系统所有者”和/或“系统用户”观点，它们一般被作为一个项目的需求分析阶段的交付成果进行构造。虽然逻辑过程模型不关心实现细节或技术，但它们可以从现有应用软件中构造出来（通过逆向工程），不过这个技术还远不如对应的逆向数据工程技术成熟和可靠。

在原始的结构化分析方法学的全盛时期，系统分析的问题分析阶段也进行过程建模。分析员将构造当前系统的物理过程模型、当前系统的逻辑模型以及目标系统的逻辑模型。每个模型都是自顶向下地构造——从最一般的模型到最细节的模型。虽然概念上合理，但是这种方式导致了过度建模和项目进度明显延迟，以至于结构化技术的领袖 Ed Yourdon 称之为“分析瘫痪”。

如今，大多数现代结构化分析策略的重点是正被开发的目标系统的逻辑模型。这些模型除了以自顶向下或者自底向上的方式构造以外，还按照一种称为事件划分的常用策略进行组织。事件划分根据业务事件和对那些事件的响应将一个系统划分成子系统。图 8-13 说明了这种用于事件驱动的过程建模策略，解释如下：

- ① 构造系统上下文数据流图，以建立初始的项目范围。在这个例子中，一页纸的数据流图仅仅显示了系统与其环境的主要接口。
- ② 绘制功能分解图，将系统划分成逻辑子系统和/或功能（对于很小的系统，这一步就省略了）。
- ③ 编译事件响应或用例清单以确定并证实系统必须提供响应的业务事件。这个清单还将描述对每个事件来说需要的或者可能的响应。
- ④ 对于每个事件，在分解图中增加一个称为事件处理器的过程。分解图现在可作为系统的概要提纲。
- ⑤ 作为备选，为每个事件构造一个事件图，并进行验证，这个简单的数据流图只显示了每个事件的事件处理器以及输入和输出。
- ⑥ 通过合并事件图构造一个或者多个系统图。这些数据流图显示了系统的“整体视图”。
- ⑦ 对需要进一步处理细节的事件过程构造基本图。这些数据流图显示了单个事件的所有基本过程、数据存储和数据流。
- ⑧ 每个基本过程的逻辑。
- ⑨ 每个基本数据流图的数据结构均使用本章前面描述的工具描述。

上面这些过程模型一起记录了一个系统的所有业务过程需求。我们将在音阶公司案例研究中演示这些技术。

系统分析得出的逻辑过程模型描述了系统的业务处理需求，而非技术方案。回忆在第 4 章中，决策分析阶段的目的是决定用技术实现需求的最佳方法。在实践中，这个决策可能已经作为一个应用架构的一部分被标准化了。例如，音阶公司应用架构要求开发团队首先决定是否可以买到一个可接受的系统。如果买不到，当前的应用架构要求内部构造的软件用微软的 Visual Basic.NET 或 C# 编写。

像所有的系统模型一样，过程模型也存储在资料库中。计算机辅助系统工程（CASE）技术为存储过程模型及其详细描述提供了资料库。大多数 CASE 产品支持计算机辅助过程建模，大多数 CASE 产品也都支持分解图和数据流图，有一些 CASE 产品还支持业务过程分析和重构。

使用 CASE 产品，你可以不用纸、笔、橡皮擦和模板就能够方便地构造出专业的、可读的过程模型，这些模型也可以被方便地修改以反映最终用户的建议。而且，大多数 CASE 产品提供了强有力的分析工具，这些分析工具可以检查模型的手工错误、完整性和一致性，有些 CASE 产品甚至可以帮助你分析数据模型的一致性、完整性和灵活性。节省时间和提高质量的潜力是巨大的。

CASE 工具也有缺点。不是所有的过程模型规则都被所有的 CASE 产品支持。所以，CASE 产品可能会强制公司采用其方法学的过程建模符号或方式，以便可以在 CASE 工具的限制范围内工作。

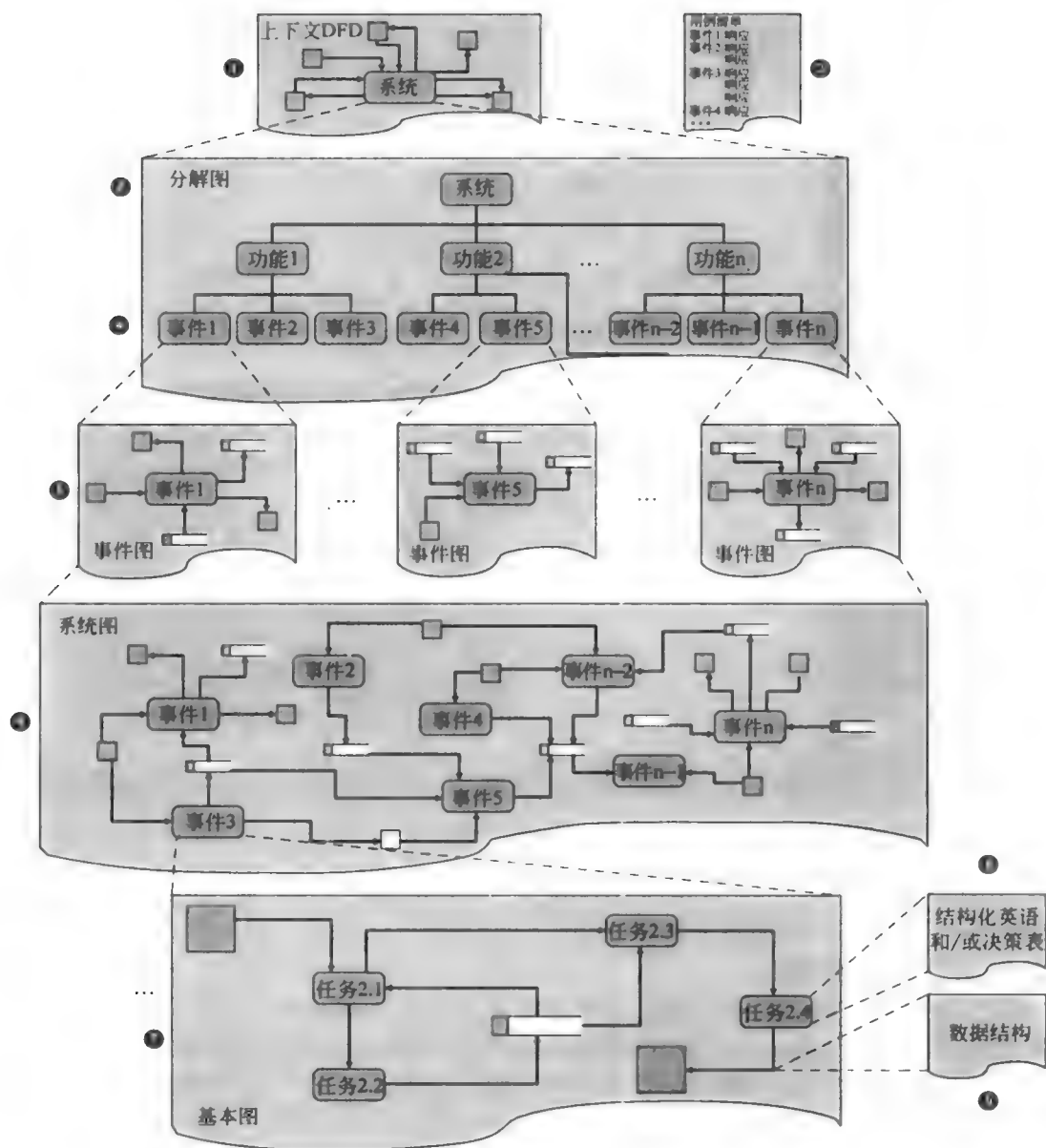


图 8-13 事件驱动的过程建模策略

本章下一节中所有的音阶公司案例研究中的过程模型都使用 Popkin 公司的 CASE 工具 System Architect 2001 构造。出于案例研究的目的，我们提供与打印机输出完全一样的打印输出，没有添加颜色，唯一的艺术处理是在打印输出中的提醒读者注意特别项目的标记。音阶公司的过程模型中的所有过程、数据流、数据存储和边界都被自动地分类存储到 System Architect 的项目资料库中（System Architect 称之为百科全书）。图 8-14 演示了用于数据建模的一些 System Architect 屏幕。

8.4 如何构造过程模型

作为一个系统分析员或者有知识的最终用户，为了建模业务过程需求，你必须知道如何绘制分解图和数据流图。我们将使用音阶娱乐俱乐部案例研究项目来讲授如何绘制这些过程模型。

我们假定项目的初始研究阶段和问题分析阶段已经完成了，而且项目团队理解了当前系统的优势、

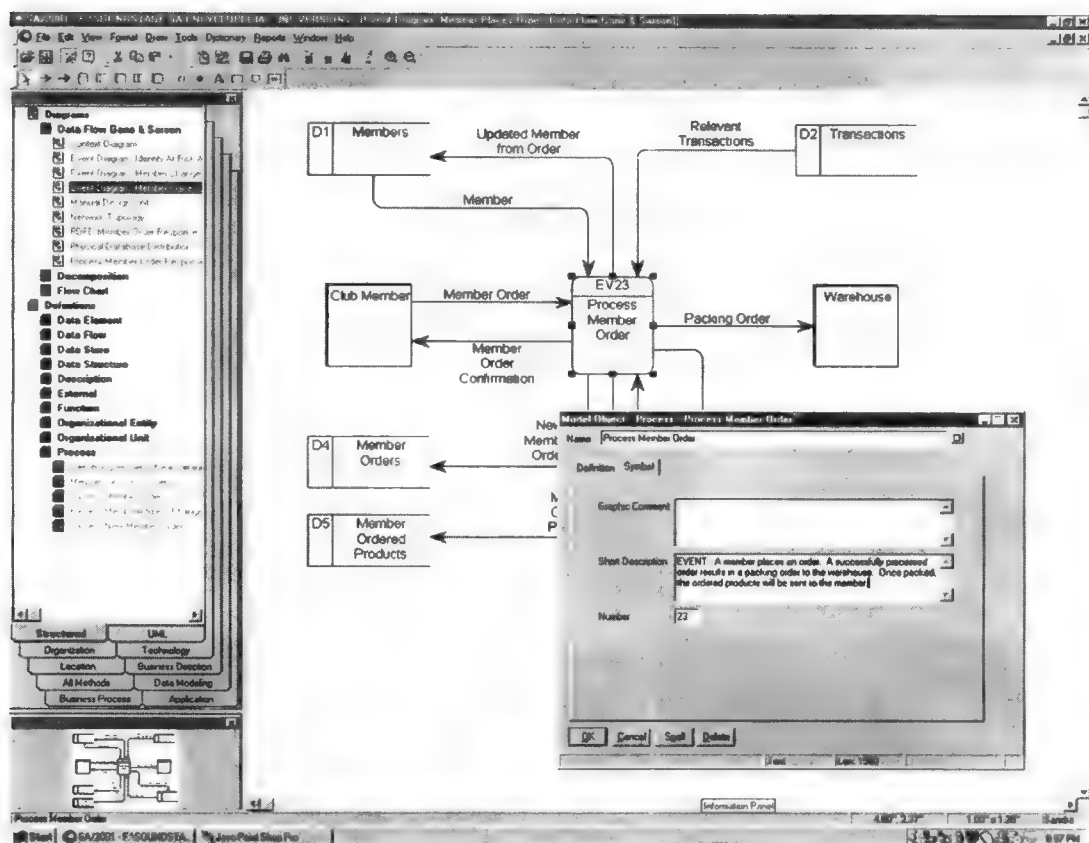


图 8-14 用于过程建模的 CASE 工具（使用 Popkin Software & Systems 公司的 System Architect 2001）

弱势、限制、问题、机会和约束。团队也已经构造了一个数据模型（见第 7 章）来记录新系统的业务数据需求，团队成员现在将构造相应的过程模型。

8.4.1 上下文数据流图

首先，我们需要记录项目的初始范围。所有的项目都有范围，项目范围定义了一个系统或应用程序准备支持业务的方面，它也定义了正被建模的系统必须如何与其他系统和企业进行交互。在信息系统框架中，项目范围定义为来自“系统所有者”视角的“通信”焦点，它使用了上下文数据流图。因为任何项目的范围总是会变化的，所以上下文图也在不断地变化。同义词是环境模型 [Yourdon, 1990]。

上下文数据流图包含了一个且仅一个过程（见图 8-15）。有时，这个过程被确定为编号“0”；但是，我们的 CASE 工具不支持这种编号。外部代理被绘制在边缘区域。数据流定义了系统与边界以及系统与外部数据存储的交互。

正如图 8-15 所描述的，系统的主要目的是处理“新订阅”以响应“订阅请求”、为产品创建“新促销商品”、通过发送要被仓库执行的“提货单”来响应“会员订单”（注意我们保持所有的数据流名称为单数）。管理人员也强调了对“各种报告”的需要。最后，对这个系统的 Web 扩展要求系统为会员提供有关订单和账号的“各种查询响应”。

8.4.2 功能分解图

功能分解图显示了一个系统的自顶向下的功能分解结构，也为我们提供了用于绘制数据流图的提纲。

图 8-16 显示了音阶公司项目的功能分解图，下面我们来研究这张图。首先，注意过程用矩形表示，而不是圆角矩形，这只是我们使用的 CASE 工具的功能分解图的一个限制，你必须适应 CASE 工具。

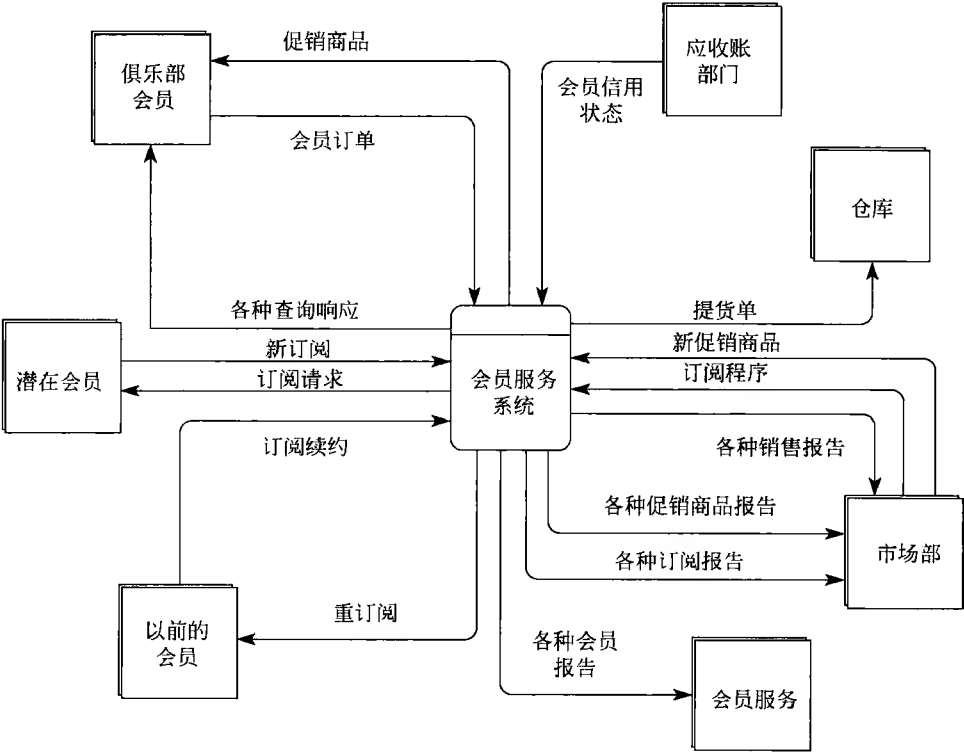


图 8-15 上下文数据流图（使用 System Architect 2001 创建）

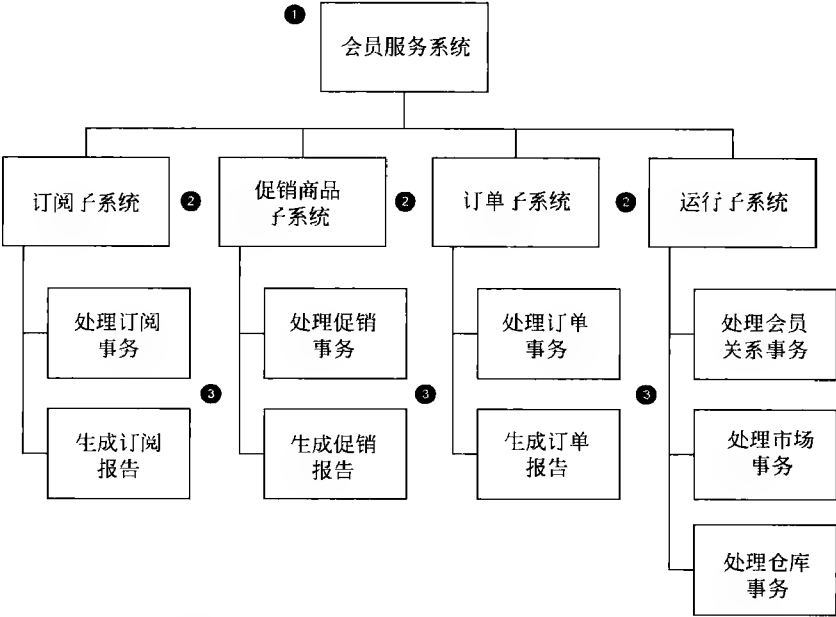


图 8-16 一个功能分解图（使用 System Architect 2001 创建）

下面是对功能分解图中逐项内容的讨论，带圆圈的数字对应图中的特定注释。

- ①根过程对应整个系统。
- ②系统最初被分解成子系统和/或功能，这些子系统和功能不需要与组织结构图上的组织部门相对应。分析员和用户越来越被要求忽略组织边界，并构造理顺过程和数据共享的交叉功能系统。
- ③我们喜欢区分系统的运行性部分和报告，以此来分解每个子系统。如果这个结构以后不起作用了，我们还可以改变它。

较大的系统可能首先被分解成子系统和功能。对于一个父过程，子过程的数量没有限制。许多作者过去常常推荐每个父过程最多只可分解为5~9个子过程，但任何这种限制都是人为的。相反，应该结构化这个系统以使它对业务来说是合理的！

把一个父过程分解成一个子过程没有意义，这没有提供任何进一步的信息。所以，如果你计划分解一个过程，至少应该将它分解成两个过程。

8.4.3 事件响应或用例清单

构建分解图后，下一步是确定系统必须响应什么业务事件，以及什么样的响应比较合适。发现事件并不难。上下文图中的有些输入与事件相关，但上下文图很少能够显示出所有的事件。从本质上说，存在以下三类事件：

- **外部事件**由外部代理引发。当这些事件发生时，就出现一个到系统的输入数据流。例如，事件“客户发出一个新订单”就以来自外部代理“客户”的输入数据流的形式而识别。
- **时序事件**以时间为基础触发过程，或者某事只是发生了。当这些事件发生时，就产生一个输入控制流。时序事件的例子可以是“提醒客户支付上次票据的时间”或者“月末”。
- **状态事件**基于系统从一个状态或条件到另一个状态或条件的转变触发过程。像时序事件一样，状态事件将用一个输入控制流表示。

信息系统通常主要响应外部事件和时序事件。状态事件通常与实时系统有关，例如电梯控制或机器人控制。

发现并确定事件和响应的更流行、更成功的方法之一是一种被称为用例的技术（见第6章），它由Ivar Jacobson博士开发。该技术根植于面向对象分析中，但可以容易地调整用于结构化分析和数据流图中。用例分析是确定并建模业务事件、谁触发事件以及系统如何响应事件的过程。

用例从用户角度确定并描述需要的系统过程。每个用例都由用户或者外部系统触发，它们称为参与者。参与者是任何需要同系统交互以交换信息的事物，因此它等价于DFD中的外部代理。

上下文数据流图确定了作为外部代理的关键参与者，它也确定了一些用例。其中的关键词是“一些”，由于上下文图仅仅显示了一个系统的主要输入和输出，所以总是会有比上下文图所描绘的更多的输入和输出——通常多得多。有些描绘的输入和输出实际上包含了许多类型和变种（例如，在上下文图中的“各种报告”）。而且，上下文图可能没有说明许多异常的输入和输出（例如错误、询问、后续等）。

图8-17演示了用例清单的一部分内容。对于每个用例，我们列出：

- 引发事件的参与者（它们将成为DFD中的外部代理）。
- 事件（它们由DFD中的某个过程处理）。
- 输入或者触发器（它们将成为DFD中的一个数据流或者控制流）。
- 所有的输出和响应（它们也将成为DFD中的数据流）。注意我们使用括号来指示时序事件。
- 输出（但注意不要暗示实现方式）。当我们使用报告这个词时，并不一定是指书面报告。注意响应包括了修改数据模型中存储的实体数据，如：创建新的实体实例、修改实体的现有实例以及删除实体实例。

一个系统的用例通常很多，这对于系统设计人员构造一个完整的响应所有业务事件的系统来说是必要的。最后考虑给每个事件分配一个在前面的步骤中绘制的分解图中确定的子系统和功能。

参考者/外部代理	事件（或用例）	触发器	响 应
市场部	制定一个新的会员关系订阅计划以吸收新会员	新会员订阅程序	生成“订阅计划确认”在数据库中创建“合同”
市场部	制定一个新的会员关系重订计划以吸引原先的会员	以前的会员重订计划	生成“订阅计划确认”在数据库中创建“合同”
市场部	为当前会员改变订阅计划（例如，延长履行时间）	订阅计划改变	生成“合同修改确认”修改数据库中的“合同”
（时间）	一个订阅计划过期	（当前日期）	生成“合同修改确认”在数据库中逻辑地删除（置空）“合同”
市场部	在达到计划的过期日期之前取消一个订阅计划	订阅计划取消	生成“合同修改确认”在数据库中逻辑地删除（置空）“合同”
会员	通过订阅加入俱乐部（按 1 美分的价格购买任意 12 张 CD，并同意在两年内按照正常价格购买 4 张以上 CD）	新订阅	生成“会员目录修改确认”在数据库中创建“会员”在数据库中创建第一个“会员订单”和“会员订购的产品”
会员	修改地址（包括电子邮件和密码）	地址修改	生成“会员目录修改确认”修改数据库中的“会员”
应收账款部门	修改会员的信用状态	信用状态修改	生成“信用目录修改确认”修改数据库中的“会员”
（时间）	市场部决定停止销售一个产品后 90 天	（当前日期）	生成“目录修改确认”在数据库中逻辑地删除（失效）“产品”
会员	挑选商品（逻辑需求由基于 Web 的信息访问视图所驱动）	产品查询	生成“目录描述”
会员	发出订单	新会员订单	生成“会员订单确认”在数据库中创建“会员订单”和“会员订购的产品”
会员	修改订单	会员订单修改请求	生成“会员订单确认”修改数据库中的“会员订单”和/或“会员订购的产品”
会员	取消订单	会员订单取消	生成“会员订单确认”在数据库中逻辑地删除“会员订单”和“会员订购的产品”
（时间）	订单处理后 90 天	（当前日期）	在数据库中实际地删除“会员订单”和“会员订购的产品”
会员	查询会员的购买历史记录（3 年时间限制）	会员购买查询	生成“会员购买历史”
（每个）俱乐部	（月末）	（当前日期）	生成“月度销售分析报告” 生成“月度会员合同例外情况分析报告” 生成“会员关系分析报告”

图 8-17 部分用例表

8.4.4 事件分解图

为了进一步在分解图中划分功能，我们增加事件处理过程（每个用例一个）到分解图中（见图 8-18）。如果整个分解图不能在一页中绘制，可以为子系统或功能增加独立的页，一个后续页中的根过程应该从前面的页中复制过来以提供交叉索引。图 8-18 只显示了“会员关系”子系统的事件过程，“促销商品”和“订单”功能的事件将位于独立的页中。

对分解图的细分程度没有必要超出事件和报告以外，这就像是写提纲一直写到包括文章最后的段落或句子。分解图（如所构造的）可作为数据流图的一个好提纲。

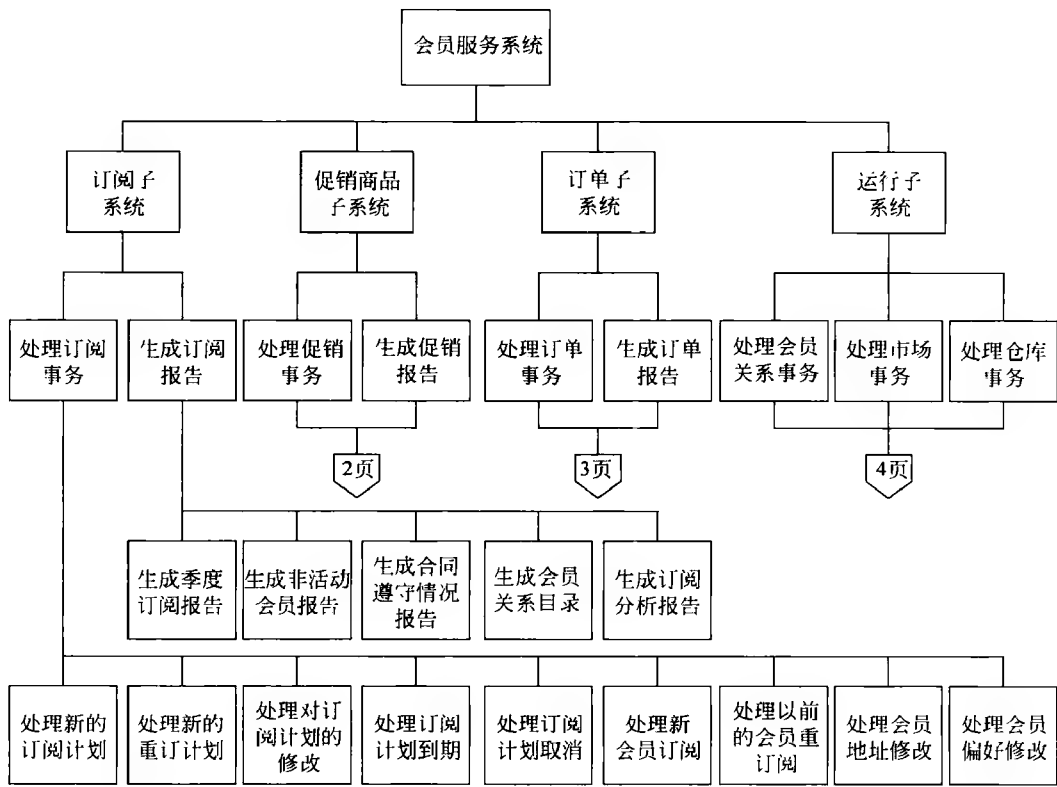


图 8-18 一个部分事件分解图（使用 System Architect 2001 创建）

8.4.5 事件图

以分解图为提纲，可以为每个事件过程绘制一个事件图。这是一个备选的但有用的步骤。事件图是一个事件的上下文图，它显示了事件的输入、输出和数据存储交互。事件图容易绘制。通过为每个过程绘制一个事件图，用户不会被系统的整体规模所吓倒。他们可以通过每个用例自己的上下文图检查用例。

在绘制任何事件图之前，列出所有可用的数据存储可能会有所帮助。因为音阶公司已经为该项目完成了数据模型，所以团队成员只需根据那个数据模型中的每个实体名称创建一个复数名词列表就可以了。检查列表中每个实体/数据存储的定义和属性是有用的。

数据存储（实体）	
合同	产品
会员	促销商品
会员订单	促销商品目录
会员订购的产品	

大多数事件图只包含一个单一的过程——过程的命名与分解图中事件的名称相同。对于每个事件，需要说明以下内容：

- 输入以及输入的来源，来源被描述为外部代理。每个输入的数据结构都应该被记录在资料库中。
- 输出以及输出的目的地，目的地被描述为外部代理。每个输出的数据结构都应该被记录在资料库中。

- 从它们那里“读取”记录的任何数据存储都必须添加到事件图中。数据流应该被添加并命名，以反映过程读取了什么数据。
- 从它们那里创建、删除或修改记录的任何数据存储都必须包含到事件图中。应该命名到数据存储的数据流，以反映修改的特性。

事件图的简单性使得这项技术成为用户和技术专家之间一种强有力的沟通工具。

音阶公司案例研究中的一份完整事件图将使本章的篇幅加倍，但不会明显提高教育价值。因此，我们将只用三个简单的例子来演示这个模型。

图 8-19 演示了一个外部事件的一个简单的事件图。大多数系统都有许多这种简单的事件图，因为所有系统都必须提供对数据存储的常规维护。

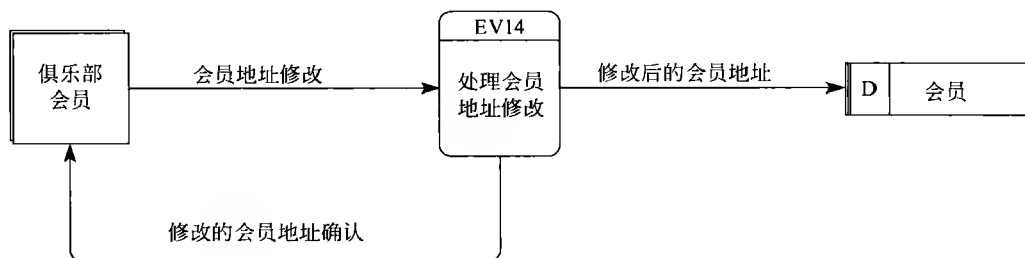


图 8-19 一个简单的数据流图（使用 System Architect 2001 创建）

图 8-20 描述了一个较复杂的外部事件，是业务事务“会员订单响应”的一个事件。注意业务事务往往会使用和修改更多的数据存储，并同外部代理有更多的交互。

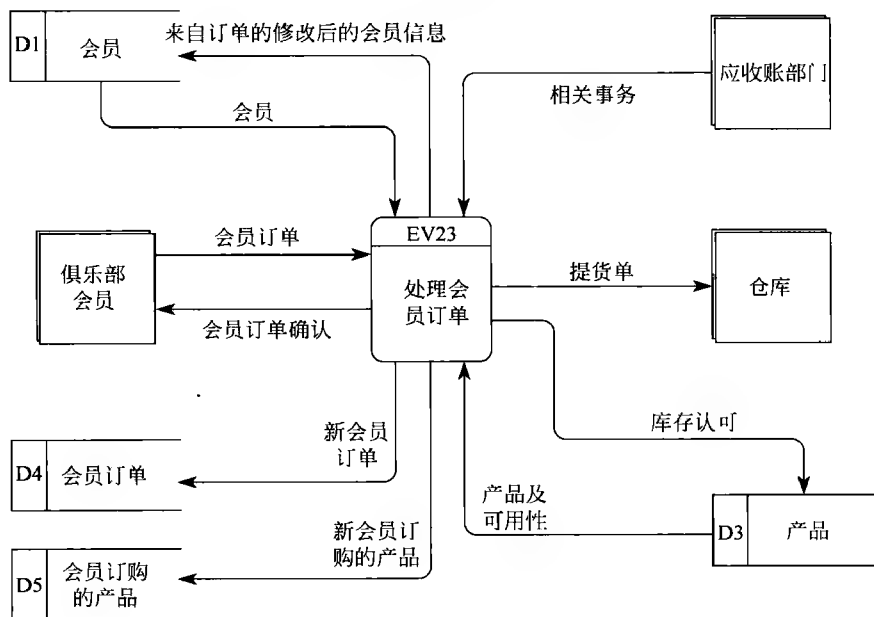


图 8-20 一个较复杂的数据流图（使用 System Architect 2001 创建）

一个事件图可以有多个过程吗？答案是可以。有些事件过程可以触发其他事件过程。在这种情况下，事件的组合应该被显示在单个事件图中。根据我们的经验，大多数事件图只有一个过程，偶尔有一个事件图具有两个或者三个过程。如果过程数量超过三个，你可能在绘制活动图（过早地），而不是事件图——换句话说，你太多地卷入了细节问题。大多数事件过程不直接互相沟通，相反，它们通

过共享的数据存储进行沟通。这使得每个事件过程只需完成它自己的工作，而不用担心其他过程的运行。

图 8-21 显示了一个时序事件的事件图。我们增加了一个外部实体“日历”或“时间”作为这个控制流的来源。

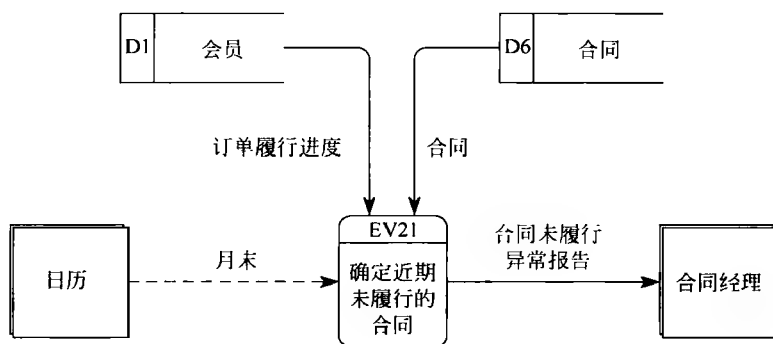


图 8-21 一个时序事件图（使用 System Architect 2001 创建）

8.4.6 系统图

事件图对用户验证系统必须提供响应的每个事件的正确性很有意义。但这些事件并不是孤立存在的，它们集合在一起定义了系统和子系统。所以，构造一个或多个显示系统或子系统中所有事件的系统图也是有用的。

系统图被形象地称为是从原始的上下文图（见图 8-15）中的单个过程“爆炸”出来的。系统图：1）在单张图中显示了系统的所有事件；2）在单张图中显示了子系统的所有事件。由于系统大小的原因，单张图可能太大。

我们的音阶公司项目是一个中等规模的项目，但它仍响应了太多的事件，以至于不能把所有过程都挤到一张图中。因此，Bob Martinez 选择为每个主要的子系统绘制一个子系统图。图 8-22 显示了“订单子系统”的子系统图，Bob 把这个子系统的所有事务和报告编写事件合并到一幅图中（如果图中内容太多，报告事件可以被忽略或者合并成组合事件）。注意系统图演示了事件过程如何真正地沟通——使用共享的数据存储。

我们现在有了一套事件图（每个业务事件一个）以及一个或多个系统/子系统图，而且事件图中的过程被合并到了系统图中。保证事件图中的每个数据流、数据存储和外部代理都表示在系统图中是十分重要的。注意，在图 8-22 中我们重复了数据存储和外部代理以尽量减少交叉线。大多数 CASE 工具都提供了工具来检查平衡错误。

在结束这个主题之前，应该介绍平衡的概念。平衡是指同步不同详细程度的数据流图以保持模型的一致性和完整性。平衡是一种质量保证技术。如果扩展一个过程成为另一个 DFD 以揭示更多的细节，平衡要求必须在子图中包含与父图中的原始过程相同的数据流和数据存储（或者它们的逻辑替代）。

8.4.7 基本图

系统图中的某些事件过程可以扩展成一个基本数据流图，以揭示更多的细节，这对较复杂的业务事务过程（例如订单处理）特别有必要。有些事件（例如报告生成）已经足够简单，所以不需要再进一步扩展。

具有较复杂的事件图的事件过程应该扩展成一个更详细的基本数据流图，如图 8-23 所示。这个基本 DFD 显示了该事件详细的过程需求，它显示了事件过程的几个基本过程，每个基本过程都是内聚的——也就是说，仅完成一件事。在基本图中，允许有流连接基本过程。

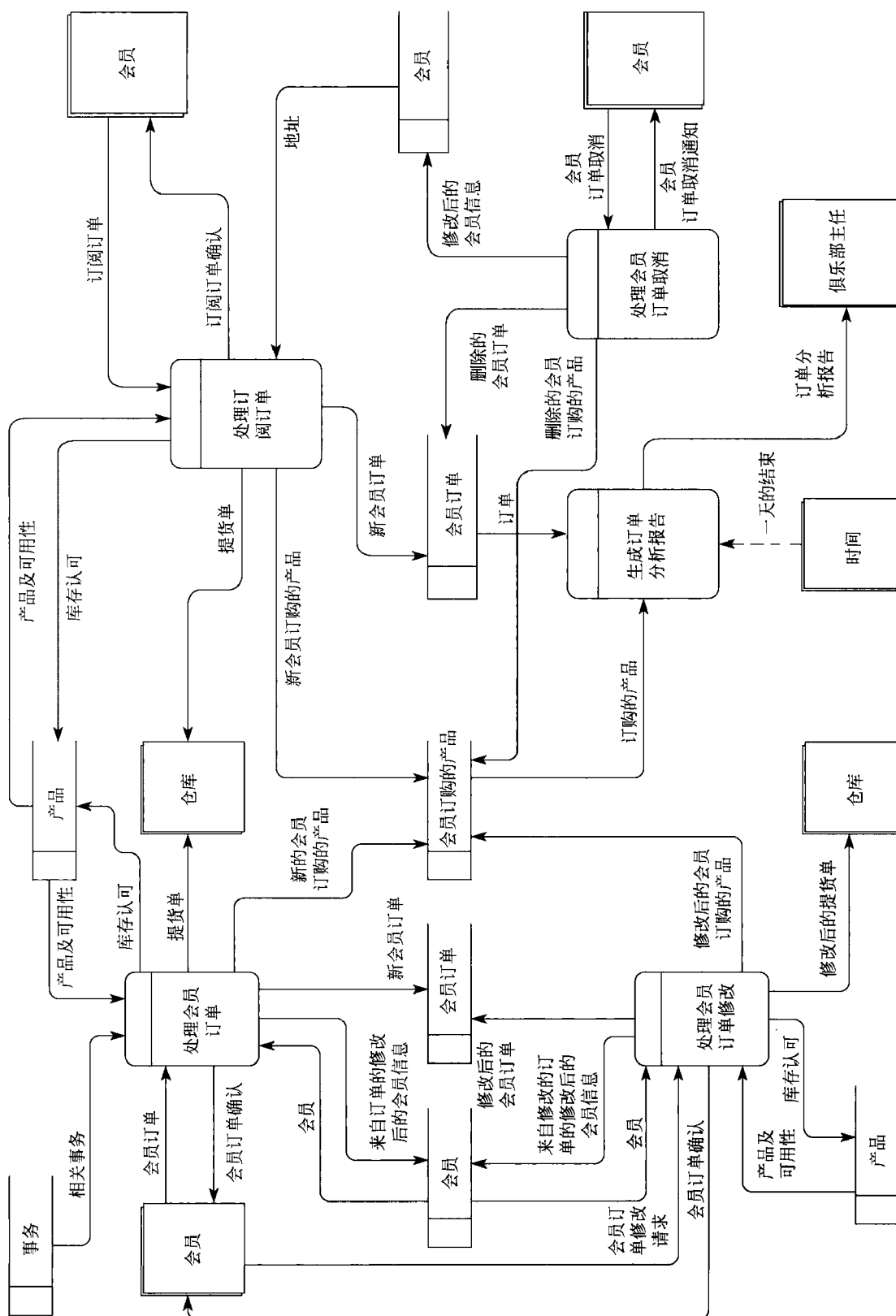


图8-22 一个系统图(使用System Architect 2001创建)

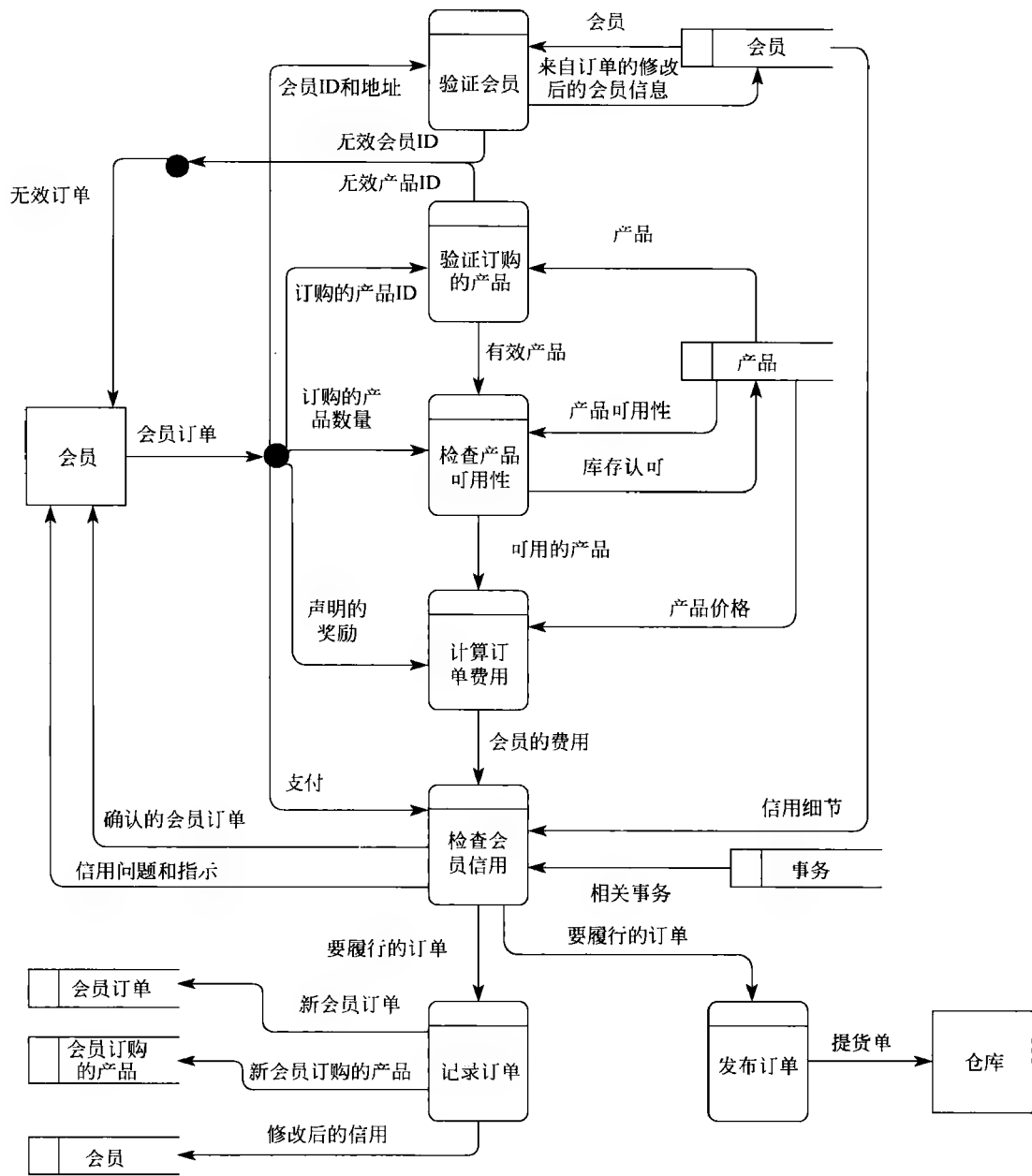


图 8-23 一个基本图（使用 System Architect 2001 创建）

当 Bob 绘制这个基本数据流图时，要在过程之间增加新的数据流。为此，他试图应用数据守恒，确保每个过程仅获得它真正需要的数据。每个数据流的数据结构都要在 CASE 工具的资料库中描述。还要注意他使用数据流连接符来分割和合并图中的相应数据流。

注意，基本 DFD 包含了一些图 8-22 中没有的新的例外数据流。当检查这个 DFD 时，想象出一个程序结构应该不难。

上下文图、系统图、事件图和基本图的组合构成了过程模型。一个工艺良好的完整过程模型可以在最终用户和计算机程序员之间有效地沟通业务需求，消除大部分系统设计、编程和实现阶段中经常出现的混淆。

8.4.8 完成规格说明

数据流图完成了，下一步要做什么呢？这取决于你选择的方法学。如果你使用纯粹的结构化分析方法学（从中导出数据流绘图），你就必须完成规格说明。为此，每个数据流、数据存储和基本过程（意味着不会进一步扩展为更详细的 DFD 的过程）必须被描述到数据资料库中。CASE 工具为这种描述提供了工具。

数据流使用本章前面解释的数据结构描述。图 8-24 演示了 System Architect 2001（音阶公司的 CASE 工具）如何用于描述一个数据流。注意这个 CASE 工具使用了类似本章描述的代数符号记法。最后，每个数据元素或属性还应该在数据字典中描述，以说明数据类型、域和默认值（见第 7 章的描述）。数据存储对应数据模型中的一个数据实体的所有实例。因此，在对应每个实体及其属性的数据字典中，对它们进行了最佳的描述（见第 7 章）。有些分析员喜欢将每个数据存储的内容转换成类似于图 8-24 中使用的关系数据结构来描述数据流。我们可以考虑这样的作业：使用数据模型的实体描述图来描述数据存储的内容。除此以外，为数据存储定义数据结构可能会产生数据模型和过程模型之间的同步错误——如果你要修改数据模型中的一个实体，你就必须记着在相应的数据存储的数据结构中进行同样的改变。这个工作量太大，除非你有一个 CASE 工具能够自动为你做这份工作。

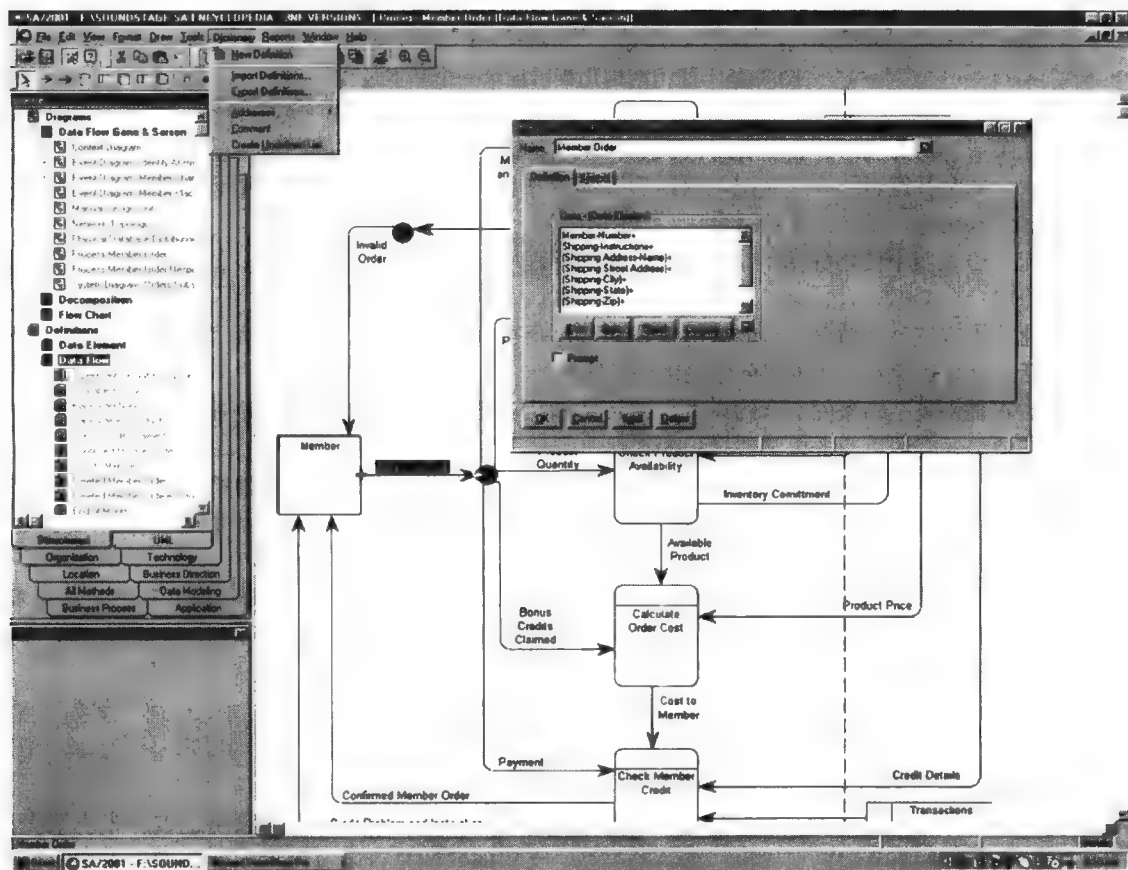


图 8-24 一个数据流（使用 System Architect 2001 创建）

过程逻辑

分解图和数据流图是对确定过程很有效的工具，但是它们并不善于显示过程内部的逻辑。我们最终将需要在数据流图中说明基本过程的详细指令。例如，考虑一个名字为“检查客户信用”的基

本过程。这个命名过程本身不足以解释“检查客户信用”所需的逻辑，所以我们需要一种有效的方法来建模基本过程的逻辑。理想情况下，逻辑模型对于与用户（用户必须验证逻辑的业务正确性）沟通和与程序员（程序员可能要用程序设计语言实现这个业务逻辑）沟通应该同样有效。

为了解决这些问题，我们的工具需要结合自然英语和编程逻辑工具的优点。**结构化英语**是一种语言和语法，结合了结构化编程和自然英语的相对优点，用于说明过程模型（例如数据流图）中的基本过程的内部逻辑。图 8-25 显示了一个结构化英语的例子（每个语句开头的数字和字符是可选的。有些最终用户喜欢使用这些字符，因为他们进一步从说明中消除了编程的“感觉”）。

1. 对于数据存储 CUSTOMER 中的每个 CUSTOMER NUMBER:
 - a. 对于匹配上面的 CUSTOMER NUMBER 的数据存储 LOANS 中的每个 LOAN:
 - 1) 保持 CUSTOMER NUMBER 的 NUMBER OF LOANS 的总数
 - 2) 保持 CUSTOMER NUMBER 的 ORIGINAL LOAN PRINCIPAL 的总数
 - 3) 保持 CUSTOMER NUMBER 的 CURRENT LOAN BALANCE 的总数
 - 4) 保持 CUSTOMER NUMBER 的 AMOUNTS PAST DUE 的总数
 - b. 如果 CUSTOMER NUMBER 的 TOTAL AMOUNTS PAST DUE 大于 100.00, 则:
 - 1) 在数据流 LOANS AT RISK 中记录 CUSTOMER NUMBER 和数据
- 否则
 - 2) 从数据流 LOANS AT RISK 中删除 CUSTOMER NUMBER 和数据

图 8-25 使用结构化英语记录一个基本过程

结构化英语不是伪代码，它不包含声明、初始化、链接之类的技术问题。但是，它确实借用了结构化程序设计的某些逻辑结构来克服英语语言在结构和精确性上的不足。请把结构化英语看作是自然英语语言和结构化程序设计语法的结合。

结构化英语说明中使用的基本结构已经被结构化程序设计使用了将近 30 年，这些基本结构总结在图 8-26 中。

结构化英语应该足够精确，以能够清楚地向程序员或用户说明需要的业务程序。但是它也应具有必要的灵活性，以避免花大量的时间争论语法问题。

许多过程是由复杂的条件组合约束的，使用结构化英语不容易表达它们，这种情况在业务策略中最常见。**策略**是一套约束业务中某些过程的规则。

在大多数公司中，策略是决策的基础。例如，一个信用卡公司必须按照各种符合州和联邦政府法规的策略（例如，财产率上限和支付下限）向持卡人收费。策略由规则构成，如果用户和系统分析员可以正确地向计算机程序员揭示那些规则，那么就能将规则转换成计算机程序。

好在我们有办法形式化策略规范说明，以及其他复杂的条件组合。这类逻辑建模工具就是**决策表**。虽然不了解它的人往往不使用它，但决策表对说明复杂的策略和决策规则很有用。图 8-27 演示了一张简单的决策表的三个部分：

- 条件段（上面的行）描述了将影响决策或策略的条件或因素。
- 行动段（下面的行）以语句的形式描述可能采取的策略行动或决策。
- 规则（列）描述了在一个特定的条件组合下采取哪个行动。

这幅图为一个杂货店描述了支票兑现卡背面的一个支票兑现策略，同样的策略也被一张决策表记录下来。三个条件影响支票兑现决策：支票的类型、支票的数额是否超过最大限额、发出支票的公司是否得到商店授权。行动（决策）是兑现支票或者不兑现支票。注意条件的每个组合定义了导致一个行动的一条规则，用一个“×”表示。

决策表和结构化英语都可以描述一个基本过程。例如，在结构化英语说明中一个合法的陈述语句可以读为：“使用决策表 LMART CHECK CASHING POLICY 决定是否兑现支票”。

结 构	例子模板																																			
顺序——无条件地执行一个步骤序列。	[第 1 步] [第 2 步] [第 n 步]																																			
简单条件步骤——如果指定的条件为真,则执行第一组步骤;否则,执行第二组步骤。如果条件只有两个可能值,则使用这个结构。(注意:第二组条件是备选的。)	if [真值条件] 则 [步骤序列或者其它条件步骤] 否则 [步骤序列或者其它条件步骤] End if																																			
复杂条件步骤——测试条件的值,然后执行相应的步骤集。 如果条件有两个以上的值,则使用这个结构。	根据 [条件] 执行以下步骤: 情况 1: if [条件] = [值], 则 [步骤序列或者其它条件步骤] 情况 2: if [条件] = [值], 则 [步骤序列或者其它条件步骤] 情况 n: if [条件] = [值], 则 [步骤序列或者其它条件步骤] End case																																			
多重条件——测试多个条件的值以确定正确的步骤集。使用决策表而不是嵌套的 if-then-else 结构来简化条件组合的复杂逻辑表示。 决策表是复杂逻辑的一种表格化表示,其中行表示条件可能的行动,列表示了条件的哪种组合导致特定的行动。	<table border="1"> <thead> <tr> <th>决策表</th><th>规则</th><th>规则</th><th>规则</th><th>规则</th></tr> </thead> <tbody> <tr> <td>[条件]</td><td>值</td><td>值</td><td>值</td><td>值</td></tr> <tr> <td>[条件]</td><td>值</td><td>值</td><td>值</td><td>值</td></tr> <tr> <td>[条件]</td><td>值</td><td>值</td><td>值</td><td>值</td></tr> <tr> <td>[步骤序列或条件步骤]</td><td>*</td><td></td><td></td><td></td></tr> <tr> <td>[步骤序列或条件步骤]</td><td></td><td>*</td><td>*</td><td></td></tr> <tr> <td>[步骤序列或条件步骤]</td><td></td><td></td><td></td><td>*</td></tr> </tbody> </table> <p>尽管它不是结构化英语结构,但可以命名决策表,并在结构化英语程序中引用。</p>	决策表	规则	规则	规则	规则	[条件]	值	值	值	值	[条件]	值	值	值	值	[条件]	值	值	值	值	[步骤序列或条件步骤]	*				[步骤序列或条件步骤]		*	*		[步骤序列或条件步骤]				*
决策表	规则	规则	规则	规则																																
[条件]	值	值	值	值																																
[条件]	值	值	值	值																																
[条件]	值	值	值	值																																
[步骤序列或条件步骤]	*																																			
[步骤序列或条件步骤]		*	*																																	
[步骤序列或条件步骤]				*																																
一个多迭代——重复步骤直到条件为假。如果无论条件的初始值是什么,步骤集都必须至少执行一次,则使用这个结构。	Repeat [以下步骤直到 [真值条件]] [步骤序列或者其它条件步骤] End-Repeat																																			
零个或多个迭代——重复步骤直到条件为假。 如果步骤集有条件地基于条件的初始值,则使用这个结构。	Do while [真值条件] [步骤序列或者其它条件步骤] End do 或者 For [真值条件] [步骤序列或者其它条件步骤] End for																																			

图 8-26 结构化英语的基本结构

基本过程可以使用结构化英语和/或决策表描述,如本章前面描述的那样。因为它们“基本的”,所以它们应该用一页纸或者更少的篇幅描述。图 8-28 演示了 System Architect 2001 如何描述一个基本过程。虽然 System Architect (像许多 CASE 工具一样)不支持决策表构造,但是绝大多数数字处理软件和电子表格软件的表格功能都可以方便地构造决策表。

一个简单的策略陈述

CHECK CASHING IDENTIFICATION CARD

A customer with check cashing privileges is entitled to cash personal checks of up to \$75.00 and payroll checks from companies pre-approved by LMART. This card is issued in accordance with the terms and conditions of the application and is subject to change without notice. This card is the property of LMART and shall be forfeited upon request of LMART.

SIGNATURE *Charles C. Parker, Jr.*
EXPIRES May 31, 2006

等价的策略决策表

条件和行动		规则1	规则2	规则3	规则4
条件段	C1:支票类型	个人支票	薪水支票	个人支票	薪水支票
	C2:支票总额小于或等于75.00美元	是	无关	不是	无关
	C3:由LMART授权的公司	无关	是	无关	不是
行动段	A1:兑现支票	×	×		
	A2:不兑现支票			×	×

图 8-27 一个简单的决策表

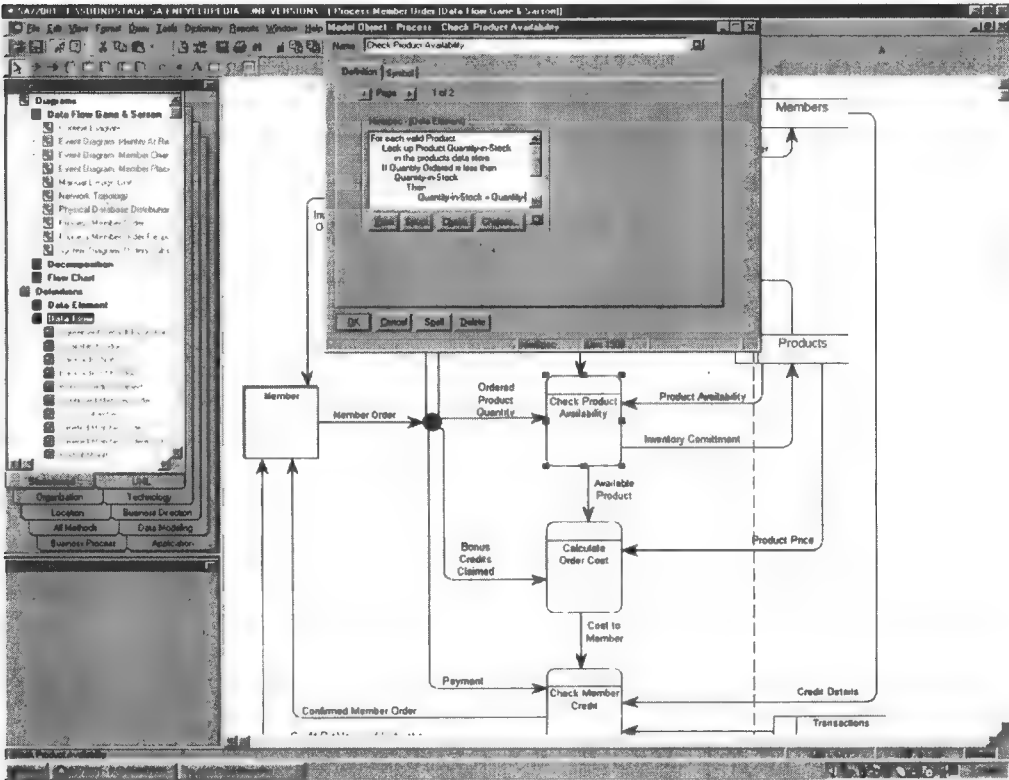


图 8-28 一个基本过程（使用 System Architect 2001 创建）

复习题

1. 为什么逻辑模型对系统分析员来说是有价值的工具?
2. 什么是数据流图, 它常见的同义词是什么?
3. 数据流图与流程图有什么区别?
4. 为什么一个系统被看作是一个过程?
5. 什么是分解, 为什么需要它? 用来描述系统分解的工具是什么?
6. 什么是三类逻辑过程?
7. 当描述数据流图和其他过程模型中的过程时常见的机械错误是什么?
8. 什么是结构化英语, 为什么在构造过程逻辑时使用它?
9. 逻辑数据流的命名规范是什么?
10. 什么是数据守恒, 为什么需要它?
11. 什么是外部代理, 为什么信息系统的外部代理会有所变化?
12. 用于系统分析的事件驱动的建模的例子有哪些?
13. 什么是用于记录信息系统的范围的过程模型, 在这个过程模型中描述了什么?

问题和练习

1. 你作为学生助手正为一家工程公司工作, 公司按照小时付费。每隔两周, 你将一份时间表格交给你的主管, 三个工作日后, 你的工资就会直接打到你的账户上。列出从你提交时间表格的时刻开始涉及的各种实体或对象、逻辑过程、数据流和数据存储。
2. 匹配第一列中的定义和第二列中的例子:

1. 结构化英语	A. 将一个系统分解成它的构件
2. 过程	B. 必须作为一个整体完成的逻辑工作单位
3. 逻辑模型	C. 逻辑建模的工具
4. 基本过程	D. 相关的和正在进行的业务活动的集合
5. 策略	E. 组织和记录系统过程的技术
6. DFD	F. 过程式说明语言
7. 决策表	G. 系统是什么或做什么的描述
8. 分解	H. 系统数据流的描述
9. 事件	I. 系统分解的描述
10. 物理模型	J. 系统为响应输入的数据流而执行的工作
11. 功能	K. 完成事件响应所需的详细的、独立的活动/任务
12. 层次图	L. 过程完成规则
13. 过程建模	M. 系统是什么或做什么, 以及它如何实现的描述

3. 在分解图中, 如何表示一个父对象的一个子对象, 如何表示一个子对象的多个父对象? 为什么分解图中的连线不像很多其他图形一样带箭头? 为什么连线不命名?
4. 考虑加利福尼亚萨克拉曼多的公共汽车道。在周一到周五的上午 6:00 ~ 10:00, 下午 3:00 ~ 7:00, 只有载客两人以上的汽车、载客一人以上的摩托车和混合 (汽油/电力) 动力车可以通行。对于所有其他车辆或者不满足上述条件的车辆, 司机收到交通罚单。这些时间以外, 就没有任何限制了。基于以上信息, 写一个供高速公路巡逻员使用的决策表。
5. 你在一个法律执行机构的调查部门的总部工作, 正在为你的总部官员开发一个案件跟踪系统以代替当前的手工系统。当收到你所在机构其他部门的调查申请表格后, 就打开案件; 没有案件是在内部初始化的。创建一个新的案件文件夹, 包括基于查询各种犯罪数据库获得的任何犯罪记录信息, 然后发送到相应的现场调查部门。当案件结束时, 总部会收到来自现场办公室的调查报告, 案件终止, 然后完整的调查报告的副本被发送到原来的部门。每周显示打开的案件、结束的案件和正在进行的案件的列表清单, 并被发送到每个现场办公室。你可以用来设置这个系统的范围和边界的几个策略是什么?
6. 基于前一个问题, 你决定这个系统的:
 - a. 净输入
 - b. 净输出
 - c. 外部代理
 - d. 外部存储
7. 现在你已经掌握了净输入、净输出、外部代理和外部存储, 绘制一个上下文数据流图。
8. 你现在正在为案件跟踪系统建模每个单元过程的逻辑, 并且已经决定用结构化英语来编写, 以便同时与用户和程序员有效地交流。对于提供给现场办公室的打开的/关闭的/进行中的案件清单报告, 编写一份结构化英语的陈述来记录保持一份实时的打开的/关闭的/进行中的案件总数的过

程。另外，你想给报告增加另一列以显示持续了6个月仍在进行中的案件数量。

- 9. 你现在准备为案件跟踪系统创建功能分解图。功能分解图的根过程是什么？一般会包括什么子系统？你会显示什么过程，它们属于什么子系统？使用这些信息来创建功能分解图。
- 10. 你的下一步是，绘制完功能分解图后创建事件响应或用例清单。对于由外部代理发起的每个事件应该有一个用例。时序事件也应该显示。对于每个外部代理，至少应该有一个用例。从一个部分用例表开始，包括事件“其他部门发出调查请求”和“现场调查组发送完成的调查报告。”还包括事件“生成打开的/关闭的/进行中的案件清单报告”。
- 11. 一个事件图等价于一个事件的上下文图。事件图包括同那个事件相关的输入、输出和数据存储交互。其目的是帮助用户关注某个事件而不会被整个系统的视图所迷惑。从案件跟踪系统中选择一个事件，绘制一张事件图。

项目和研究

- 1. 假设你正在为一个企业启动一个项目的工作，那个企业在设计系统中从来没有使用过任何建模技术或工具（是的，这很难想象，但确实存在）。你的经理不愿意改变他们一直做事的方式。写一份关于为什么系统建模值得花时间和投入资源的一到两页的问题报告（或者用一份 PowerPoint 幻灯片代替）。
- 2. 本书使用了 Gane 和 Sarson 的过程建模符号，将它们同 DeMarco/Yourdon 和 SSADM/IDEFO 过程建模方法使用的符号进行比较。研究至少两种其他过程建模方法，然后比较它们：
 - a. 你找到了什么其他过程建模方法？
 - b. 如果存在，除了符号记法不同以外，其他过程建模方法还有什么重要的区别？
 - c. 有什么相似点？
 - d. 你的企业使用哪种符号记法？
 - e. 如果要你给你的企业推荐其中的一种方法，你会选择哪一种？为什么？
- 3. 直到最近，杂志和期刊都还只有印刷版本的。出版者现在提供越来越多既有传统的印刷版本，也有数字格式的，可以从因特网上下载的期刊。考虑这两种方法涉及的过程：
 - a. 创建一个高层数据流图，描述通过邮件更新对一份杂志的印刷版本的订阅信息的典型传统方法。

12. 匹配第 1 列中的定义或例子和第 2 列中的术语：

A. 最小的有意义的数据段	1. 域
B. 相似的数据流的组合	2. 连接
C. 用数据结构的形式表达	3. 外部代理
D. 被监视的条件	4. 事件分解
E. 一个属性的合法值	5. 组合数据流
F. “饥饿过程”	6. 控制流
G. 来自一个过程的数据输出或者到这个过程的数据输入	7. 数据分解
H. 可以存储在一个属性中的数据类型	8. 数据属性
I. 组成一个数据流的数据属性的排列	9. 数据守恒
J. 同一个系统交互的外部实体	10. 数据流
K. 指定数据流是某个类型的唯一实例的符号	11. 数据结构
L. 静止的数据	12. 数据类型
M. 根据业务事件将系统分解成子系统	13. 数据存储

- b. 创建另一个高层数据流图，描述通过因特网更新对一份杂志的数字格式版本的订阅信息的过程。
- c. 这两张图有什么基本区别？
- d. 基于数据流图，在更新杂志订阅方面哪种格式更有效？你认为从订阅人的角度出发答案是一样的吗？从出版者的角度呢？
- e. 从接收和阅读杂志角度呢？从你自己的角度来看，用数字格式出版的杂志与传统印刷版本相比，有哪些优缺点？
- 4. 在 1978 年，Tom DeMarco 编著了被认为是结构化系统分析方法的经典著作——《结构化分析和系统说明》。James Wetherbe 被认为是关于“系统概念和系统思想作为系统分析和设计原理的一部分的最强有力的宣传者”的著作者，编著了大量的关于“系统思想”的文章和书籍。Edward Yourdon 是系统设计领域中一位广为人知的领袖，他由于在 1989 年的《现代结构化分析》一书中形式化了事件驱动的方法而出名。在因特网上搜索他们的站点，查找这三位系统分析和设计领域的领军人物最近的文章和/或书籍。
 - a. 你找到了哪些文章和/或网站？
 - b. 描述一下他们最近的工作。
 - c. 把这些作者对系统分析和设计的观点进行比较；特别是，你把他们的方法看作是互补的，

还是相对立的？

- d. 在系统分析和设计方面，他们预测有什么正在出现的变化或趋势？
 - e. 你觉得他们仍代表着系统分析和设计的主流思想吗？为什么？
5. 了解你的学校或企业使用的不同的信息系统。找

小型案例

1. 去一个你选择的小型公司。这个企业做什么？写一份一到两页的文章描述这个企业及其现有的系统。然后为现有系统绘制上下文层面的图和系统层面的图。你在当前系统中看到了低效率的或有弱点的内容了吗？并描述一下。
2. 在前一个案例中，你（在高层）记录了你选择的企业的现有系统。现在描述你认为对这个企业来说合适的系统。考虑效率、从一个部门到另一个部门的信息流等。存在以前的系统没有利用的持续的信息吗？你的新系统如何为企业带来优势？用两页纸的文章记录这个优势，以及上下文层面的图和系统层面的图。
3. 在第1章中，你涉及以下问题：政府服务部门承受着他们拥有和处理的大量数据的沉重负担。同来自某个服务部门的一些人交谈，并写一篇短文。服务部门必须要详细审查他们处理的大量数据，例如：失踪人口、儿童保护服务、DMV 和

到一个具有不完整的和/或过期的文档的系统。（在大多数企业中这并不难！）使用本书讲述的数据流图、数据结构和其他过程模型更新和完善这个系统的文档。使用 Gane 和 Sarson 的符号记法，除非你的企业支持另外一种不同的符号记法。

跟踪犯罪嫌疑人。你应该包括，但不限于以下题目：

- a. 这个部门（或人）的工作是什么？
- b. 他们收集和分析哪种数据？
- c. 他们对数据进行何种分析？
- d. 他们收集多少信息，从谁那收集，使用什么程序收集？

在这个练习中，利用你从交谈中收集的信息，绘制一张那个部门的现有系统的完整的 DFD。你可能需要返回去同你的联系人再次交谈，或者获取表格、报告等，以便你能对数据流有一个完整的视图。

4. 在前一个案例中，你记录了一个政府部门的现有信息系统。现在考虑信息流应该是什么样子的。在一篇短文中讨论现有的信息流，以及你认为在系统中应该有哪些信息。你认为的和实际存在的有很大的差别吗？

团队和个人练习

1. 个人/班级圆桌讨论：据说信息就是力量，经济发达国家、个人和公司等与不发达的之间的差异就在对信息的利用和控制上。信息的价值是什么？它如何影响国家和公司的能力和竞争力？你对信息的访问和使用影响到你在相关工作中成功的竞争力了吗？
2. 个人/班级圆桌讨论：假设一个软件公司为一个公司开发一个十分复杂和创新的软件包。这个公司想拥有该软件的版权（包括修改和销售），但

软件公司也想拥有软件的版权。寻找在现实生活中的一个例子。你认为谁应该拥有这个软件？为什么？法律支持哪一边？为什么？

3. 圆桌讨论：Tech start-ups 在 20 世纪 90 年代后期被认为具有延长的工作时间。也就是，很多时候程序员会被要求每周工作 60 ~ 80 个小时而没有加班工资。查找你所在州的劳动法，然后在班级进行圆桌形式的讨论，讨论劳动法和公司文化对技术雇员的影响。

使用 UML 进行面向对象分析和建模

本章概述和学习目标

本章是介绍系统开发之面向对象工具和技术的一个章节中的第1章，重点介绍系统分析期间的对象建模，具体包括如下内容：

- 定义对象建模并解释其优势所在。
- 认识并理解对象模型的基本概念和结构。
- 定义 UML 及其各种模型图。
- 将业务需求用例模型转换成系统分析用例模型。
- 构造活动图。
- 探讨对象和类以及它们之间的关系。
- 构造类图。

本章关键术语

面向对象分析（Object-Oriented Analysis, OOA）技术用于：1）研究现有对象，看能否复用它们或者调整它们用于新的用途；2）定义各种新对象和修改后的对象，它们将与现有对象一起组合成一个有用的企业计算应用系统。

对象建模（object modeling）是一种用于辨识系统环境中的对象和这些对象之间关系的技术。

统一建模语言（Unified Modeling Language）是一套建模规则，它使用对象说明或描述软件系统。

对象实例（object instance）由描述特定的人、地点、事物或者事件的属性值构成。

行为（behavior）是指对象可以做的事情，以及在对象数据（或属性）上执行的功能。在面向对象环境中，对象的行为通常称为方法、操作或者服务（我们可能在讨论中交叉地使用这些词汇）。

封装（encapsulation）是几项内容一起打包成一个单元（也称为信息隐藏）。

对象类（object class）是共享相同属性和行为的对象集合。类有时称为对象类。

继承（inheritance）是指在一个对象类中定义的方法和/或属性可以被另一个对象类继承或复用。

泛化/特化（generalization/specialization）是一种技术，其中几类对象类的公共属性和行为被组合成类，称为超类。超类的属性和方法然后被那些对象类（子类）继承。

对象/类关系（object class relationship）是一种存在于一个或多个对象/类之间的自然业务联系。

多重性（multiplicity）定义一个对象/类对应相关对象/类的一个实例关联可能的最小出现次数和最大出现次数。

聚合（aggregation）是一种关系，其中一个较大的“整体”类包含一个或多个较小的“部分”类。相反，一个较小的“部分”类是一个较大的“整体”类的一部分。

合成（composition）是一种聚合关系，其中“整体”负责其“部分”的创建和销毁，如果“整体”不存在了，“部分”也将不存在。

消息（message）是当一个对象调用另一个对象的方法（行为）以请求信息或者某些动作时发生的通信。

多态性（polymorphism）是指“多种形式”，意味着不同的对象可以以不同的形式响应同样的消息。

重载（override）是一种技术，其中子类使用它自己的属性或行为，而非从父类继承的属性或行为。

系统分析用例（system analysis use case）是记录系统用户和系统交互的用例，描述什么需求需要高度细化，但仍没有太多的实现细节和约束。

系统顺序图 (system sequence diagram) 是一幅描述角色和系统在用例场景下交互的图形。

类图 (class diagram) 是系统静态对象结构的图形描述, 显示了构成系统的对象类, 以及这些对象类之间的关系。

持续类 (persistent class) 是描述存活期超过创建它的程序执行时间的对象的类。

临时对象类 (transient object class) 是描述由程序临时创建并只在程序执行期间存在的对象的类。

9.1 面向对象分析简介

面向对象编程语言 (例如 Java 和 .NET 语言) 变得很流行, 原因是面向对象编程可以促进更好的代码复用以降低编程费用。另外, 面向对象方法更适合于由地理上分散的程序员小组协作开发一个集成系统的项目。每个团队可以负责开发独立的程序代码片断, 实现一个或几个具有定义的接口的对象。我们将在后面了解更多有关对象的知识。

面向对象编程方法需要**面向对象分析** (OOA) 和面向对象设计技术。有些面向对象图形, 例如类图 (在本章讲述) 和顺序图 (在第 17 章讲述) 将不适用, 除非系统在一个面向对象的环境中开发。其他的为面向对象分析和设计开发的图形可以用于任何种类的开发环境。例如, 用例现在既用在面向对象分析中, 也用在传统的结构化分析中。活动图 (在本章讲述) 和部署图 (在第 17 章讲述) 尽管是为面向对象分析和设计开发的, 但也可用于任何种类的方法中。

9.2 对象建模的系统概念

面向对象方法的核心是一种称为**对象建模**的技术。对象建模技术要求使用完全不同于数据建模和过程建模的方法和图形记号。面向对象分析涉及到定义信息系统的静态和动态行为模型, 而非定义数据和过程模型 (这是传统开发方法的目标)。这些 OOA 概念对有经验的开发人员来说是一个巨大的挑战, 他们必须重新认识他们一直以传统方式看待的系统。现今最流行的方法采用一种标准对象建模语言 (称为统一建模语言, 简称 UML)。

9.2.1 对象、属性、方法和封装

用于系统开发的面向对象方法基于这样一个概念: 对象存在于系统的环境中, 对象随处可见。以周围环境为例, 请环顾一下四周, 在周围环境中存在什么对象? 也许你看到了一扇门、一扇窗户或者房间本身。那么书呢? 它是一个对象, 你正在读的书页也是对象。也许你还有一个练习册, 它也是一个对象。如果房间里还有其他人, 他们也都是对象。你可能还看到一部电话、一把椅子或者一张桌子。所有这些都是对象, 在周围环境中, 它们都是明显可见的。

考虑一下韦氏辞典中的对象定义: “某种存在的, 或者能被看到、触摸或以其他方式感觉到的事物。”

前面提到的对象都是可以看见或者触摸到的对象, 但你可以感觉的对象是什么呢? 也许你正在等一个电话, 那个电话就是你正在感觉的某种事物。你可能在等着开一个会议, 同样地, 那个会议是你辨识、同它发生关系和预期的某种东西, 尽管你不能实实在在地看到会议。因此, 按照韦氏辞典, 一个预期的电话或会议都可以看作一个对象。

前面的例子是一些可能存在于你周围环境中的对象。同样, 在用于系统开发的面向对象方法中, 辨识那些存在于系统环境中的对象很重要。但是那些对象不仅仅被看作简单的“某种存在的, 或者能被看到、触摸或以其他方式感觉到的事物。”在用于系统开发的面向对象方法中, 对象的定义参见 4.2.1 节。

这个定义中有三个部分需要仔细考察。首先, 让我们考虑某种事物这个词。某种事物可以特征化为一类对象, 它们很像我们在你当前环境中辨识的对象。对象的类型可以包括人、地点、事物或者事件。雇员、客户、供应商和学生都是人对象的例子; 特定的仓库、办公室、建筑物和房间都是地点对象的例子; 事物对象的例子包括产品、车辆、设备、录像带或者显示在用户显示器上的一个窗口; 事件对象的例子包括订单、支付、发票、应用、注册和预约。

现在让我们考虑定义中的数据部分。在面向对象界，定义中的这个部分是指属性（参见 7.2.2 节）。

例如，我们可能关心对象“客户”的下列属性：客户编号、姓、名、家庭住址、单位地址、客户类型、住宅电话、单位电话、信用上限、可用的信用、账号余额和账号状态。在现实中，可能有许多客户对象，对于他们来说，我们将对他们的这些属性感兴趣。每个单独的客户都被称为一个对象实例。例如，对于每个客户，属性将指定那个特定客户的值——例如，412209、Lonnie、Bentley、2625 Darwin Drive、West Lafayette、Indiana、47906 等。考虑你所处的环境：也许房间里还有另一个人，你们中的每一个都代表人对象的一个实例，你们中的每一个都可以按照一些公共属性描述，例如：姓、社会保险号、电话号码和地址。

现在让我们考虑对象定义中的最后一部分——对象的行为。这代表了看待对象的一种非常不同的方式。当你看到周围环境中的门对象时，你可能仅仅看到一个不能思考的静止对象——几乎很少执行什么动作。在用于系统开发的面向对象方法中，门对象可以同假定能够在其上执行的行为相关联。例如，门可以打开，可以关闭，可以锁上，或者可以开锁。所有这些行为都与门对象相关，并且由门对象实现，而不是由其他对象实现。

另一个重要的面向对象原理是：对象单独地负责执行任何在其数据（或属性）上操作的功能或者行为，例如：只有你（一个对象）可以修改（行为）你的名字和家庭住址（你的属性）。这引出了对象的一个重要概念，即封装。对象的属性和行为都被封装到一起作为那个对象的一部分。访问或修改对象属性只能通过该对象的行为来实现。

在面向对象开发中，经常需要绘制描述对象的模型。下面介绍用来表示对象的建模符号。图 9-1a 展示了两个对象实例，对象实例使用包含对象实例名称的矩形表示。对象实例名字由唯一标示对象的属性值、冒号以及对象所属的类的名称构成。整个名字居中显示，并用下划线标出。在图 9-1a 中，属性 CUSTOMER NUMBER（它的值是 412209）唯一地标识了 CUSTOMER 的这个实例。因此，412209 是这个对象实例的名字，而 CUSTOMER 是其分类。另外，对象实例也可以如图 9-1b 这样绘制，对象实例的属性值记录在符号中，并同对象名称用一条线分隔开。

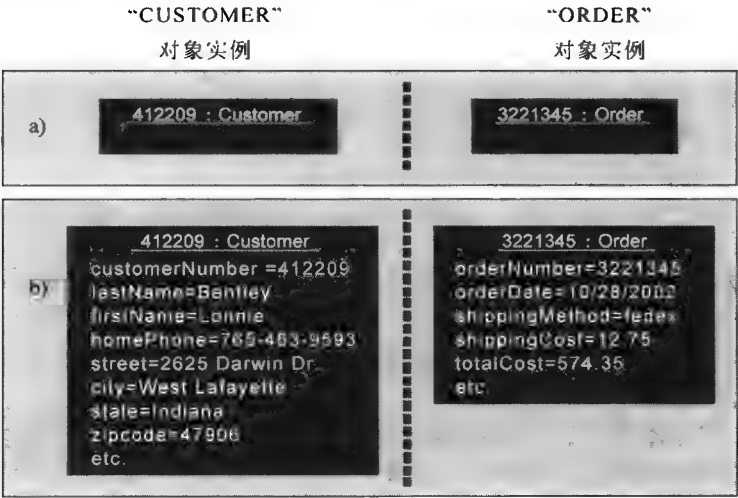


图 9-1 对象实例

9.2.2 类、泛化和特化

对象建模的另一个重要概念是对象分成类的概念。

以你周围环境中的一些对象为例。对你来说，把《系统分析和设计》课本和另一本课本（例如《程序设计导论》）归类为书是很自然的（见图 9-2a）。两个对象都表示具有类似属性和行为的事物

对象，类似的属性可能是：ISBN NUMBER（ISBN 编号）、TITLE（书名）、COPYRIGHT DATE（版权日期）、EDITION（编辑）等。同样地，它们具有类似的行为，例如能够打开（OPEN）和关闭（CLOSE）。你身边环境中可能还有几个其他对象，同样可以根据其相似性进行归类。例如，你和房间中的其他人可以归类为 PERSON（人）。

对象建模中如何使用UML符号表示类？如图9-2b所示，类的符号与对象的符号很相似，差别是属性的值被省略了，而且类名没有下划线。另外，类的符号中可以包含行为的列表。如图9-2b所示，为了便于显示包含大量类符号的图形，有时在绘制类时不列出行为和属性。大多数对象建模工具支持这个功能，以便将模型定制成你喜欢的样子。

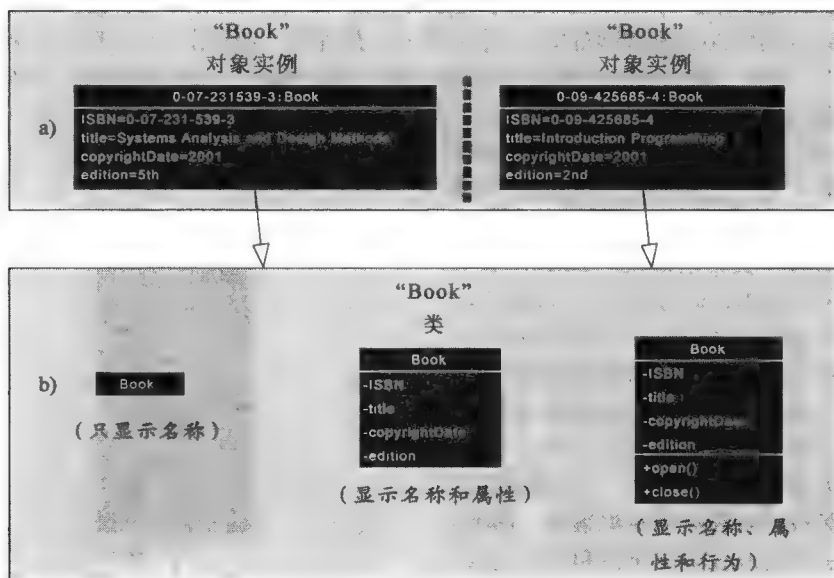


图9-2 在UML中表示类

我们还可以认识对象的子类（见图9-3a）。例如，房间中的有些人可能被归类为 STUDENT（学生），其他人被归类为 TEACHER（教师），因此，学生对象类和教师对象类就是人对象类的成员。当确定类的层次时，可以应用继承的概念。

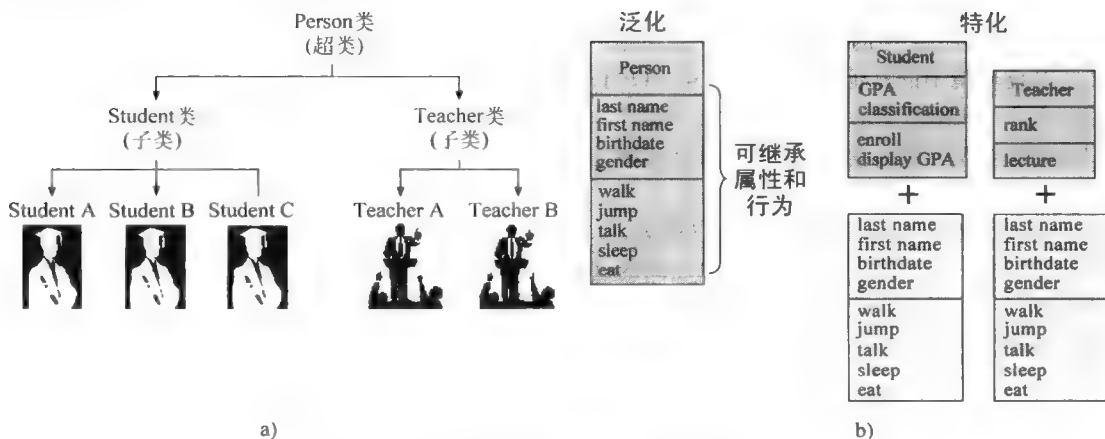


图9-3 对象类之间的超类和子类关系

发现和探索对象/类之间的共性的方法称为泛化/特化。在图 9-3b 中, 类 STUDENT 和 TEACHER 包含, 对他们来说, 特有的属性和行为 (使得它们更特化), 但它们也具有从 PERSON 类继承的泛化的属性和行为。

在我们的例子中, 对象类“人”称为超类 (或泛化类), 反之“学生”类和“教师”类称为子类 (或者特化类)。超类与子类之间具有一个或多个一对一关系, 在我们的例子中, 任何一个人 (对象实例) 将最多是一位老师, 或者一位学生, 或者两者都是。而且, 任何一个老师都将只是一个人。由于使用语句表述关系的缘故, 这些关系有时被称为“is a”关系。例如, “学生是一类人”或者“教师是一类人”。

在面向对象系统开发中, 对象按照类和子类进行分类。识别类很有好处。例如, 如果一个新属性 (性别) 需要加入到学生对象和教师对象中, 但由于这个属性对二者是公共的, 所以这个属性可以一次性加入到类“人”中——隐含着类中的两个对象都共享了那个属性。如果考虑到程序维护, 隐含就更重要了, 程序维护现在只需要简单地在—个地方修改就可以了。例如, 假设属性“LAST NAME”目前为 15 个字节长度, 通过分析我们发现属性“LAST NAME”的长度要超过 15 个字符。由于我们需要修改“LAST NAME”的长度到 25, 以便容纳所有姓的整个值。通过继承, 我们仅仅在“PERSON”类中进行一次修改。如果没有继承, 我们就不得不既修改“STUDENT”类, 也修改“TEACHER”类。前面的例子很简单, 但考虑一个可能包含几十个类上百个属性和行为的大型应用程序, 只在一处修改所节省的时间和费用就很显著了。

如何使用 UML 符号描述泛化/特化 (超类和子类)? 图 9-4 演示了如何描述人、学生和教师对象类之间的超类、子类关系。人对象的所有属性和行为都被学生和教师对象继承, 而那些唯一地应用于学生或教师的属性和行为直接记录在子类符号中。

9.2.3 对象/类关系

对象和类不是孤立存在的。它们表示的事物相互作用, 并且相互影响, 以便支持业务任务。因此, 对象/类关系是必然的。例如, 你通过阅读与本书交互, 通过使用与电话交互, 也许还通过交谈与房间中的其他人交互。类似地, 对象与系统环境中的其他对象进行交互。以可能存在于一个典型信息系统中的对象类“客户”

和“订单”为例, 我们就客户和订单如何关联 (或者交互) 做出如下的业务推断:

- 一个客户提交零个或多个订单。
- 一个订单由一个且仅由一个客户提交。

可以用图形的方式说明客户和订单之间的这种关系, 如图 9-5a 所示, 连线表示了类之间的关系。UML 称这条线为关联, 我们将在本章剩下的部分统一使用这个词。动词词组描述了关联。注意所有的关联隐含地都是双向的, 意味着它们可以在两个方向上解释 (如上面的业务推断所建议的那样)。

图 9-5a 也显示了每个关联的复杂度或维度。例如, 在上面的业务推断中, 我们还必须回答下列问题:

- 对应每个订单实例必须存在一个客户实例吗? (是的)
- 对应每个客户实例必须存在一个订单实例吗? (不是)
- 对应每个客户实例可能存在多少个订单实例? (许多)

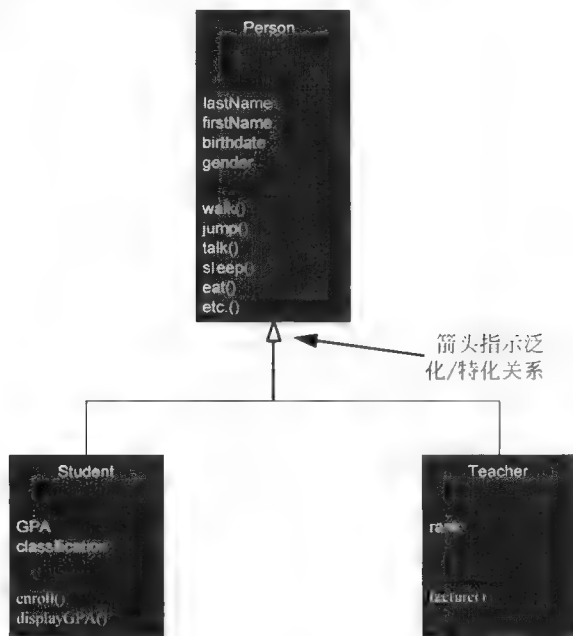


图 9-4 使用 UML 表示泛化/特化关系



图 9-5 对象/类关联和多重性符号

- 对应每个订单实例可能存在多少个客户实例？（一个）

我们称这个概念为**多重性**。因为所有关联都是双向的，意味着“客户”类知道“订单”类，“订单”类也知道“客户”类，所以对于每个关系，多重性必须在两个方向上定义。用于类之间多重性的UML图形记号如图9-5b所示。如果你已经学过了第7章的数据建模，你将意识到多重性基本上与基数是同样的概念。记号不同，但关系基本上相同。

有些对象由其他对象构成。例如，如果你在因特网上购买了一些东西，你的一个订单将由多个项目构成（CD、DVD、书等）。另一个例子是俱乐部，它由几个俱乐部成员构成；一台计算机包括一个机箱、CPU、主板、电源等。这种关系称为**聚合**，用“whole-part”或者“is part of”表示。

合成是一种强形式的聚合。考虑组成部件这个词。在合成中，“整体”完全负责“部分”的创建和销毁，“部分”只能与一个“整体”关联。俱乐部和俱乐部成员之间的关系不是合成关系，因为俱乐部成员拥有俱乐部以外的生活，而且实际上可以属于多个俱乐部。但因特网订单和订单项目是合成关系。如果你取消订单，订单中所有的项目都会被取消。在整体上执行的行为也将在其所有部分上执行。例如，如果我们打印订单，订单的所有项目也会被自动打印。

在UML的早期版本，用空心菱形绘制聚合，菱形连接到“整体”对象类，如图9-6a所示。注意必须为关系的每一边设置多重性。

用实心菱形绘制合成，如图9-6b所示。因为每个“部分”只属于一个“整体”，所以只需为“部分”设置多重性。图9-6b也展示了多级合成。书由章节构成，章节又由页面构成，等等。

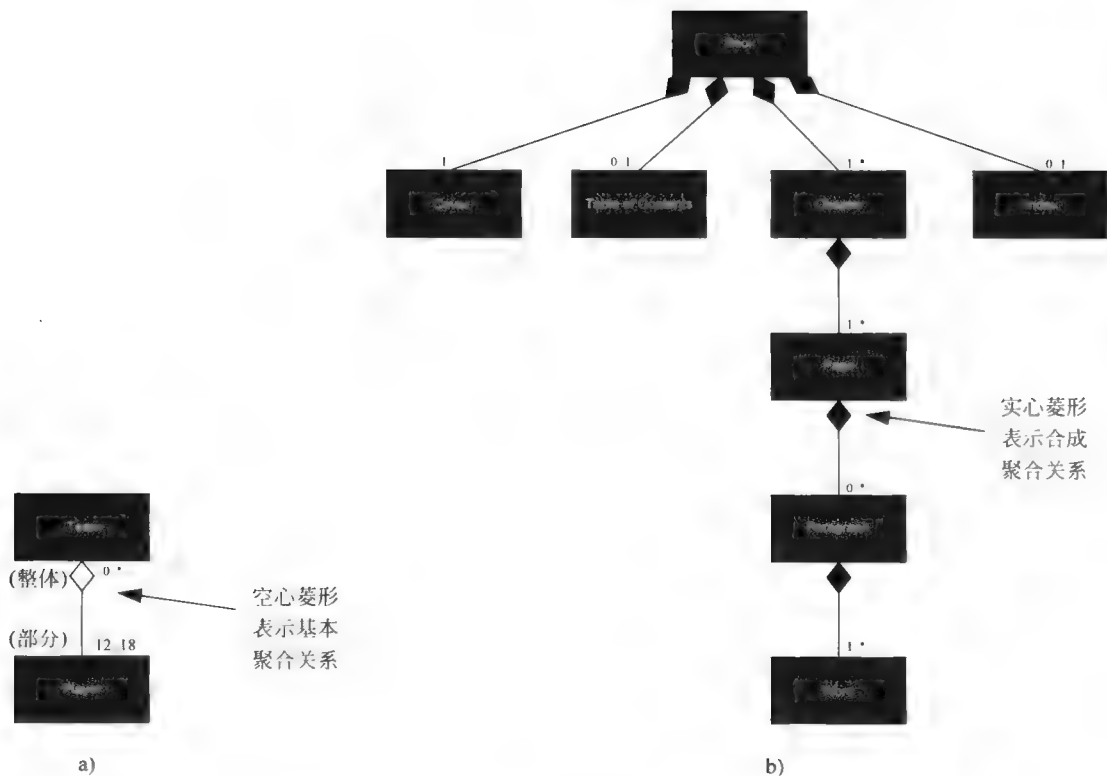


图 9-6 聚合关系

在 UML 2.0 中, 聚合的记号被取消了。为什么? 虽然合成关系在编程中确实有明确的区别, 但聚合关系总是不太区别。例如, 俱乐部和俱乐部成员之间的关系不能简单的是独立对象类之间一个或多个关联关系吗? 因此, 一些实践者认为聚合 (弱形式) 在任何实际的意义上没有基本意义。

9.2.4 消息和消息发送

对象/类通过传递消息进行交互或者“沟通”。请回顾一下封装的概念: 对象是属性和行为的封装。只有对象可以执行自己的行为, 并操作自己的数据。

以前面提到的客户对象和订单对象为例。客户对象检查订单的当前状态, 通过调用订单对象的显示状态行为 (访问并显示订单状态属性的行为), 发送一条消息给订单对象。

对象发送消息并不需要知道接收消息的对象内部是如何组织的或者行为是如何实现的, 只要知道它响应正确定义的消息请求就可以了。图 9-7 演示了这个消息概念。消息只能在有关联的两个对象之间发送。在第 17 章中将学到如何记录消息。

9.2.5 多态性

一个同消息有密切关系的重要概念是**多态性**。以周围环境中的窗户 (WINDOW) 和门对象为例, 这两个对象都有一个可以执行的公共行为: 关闭。但是门对象执行这个行为的方式可能完全不同于窗户对象执行这个行为的方式: 门“转动地关上”, 而窗户“滑下来关上”。因此, 行为“关闭”可能有两种不同的形式。更进一步, 考虑窗户对象, 实际上不是所有窗户对象都可以以相同方式实现关闭行为。有些窗户对象, 像门对象一样, “转动地关上”! 所以, 甚至在一个给定的对象类内部, 关闭行为也可能采取不同形式。

当超类的行为需要被子类的行为**重载**时, 就需要在面向对象应用中使用多态性。请看图 9-8 的泛化/特化关系, 雇员 (EMPLOYEE) 类含有一个称为“计算报酬”的行为, 用来计算每个雇员应该支付多

少报酬。因为全职雇员和兼职雇员报酬不同（全职雇员受到年薪，兼职雇员按工作的小时数付费），两个行为需要执行不同的计算。但由于多态性，行为可以采用同样的名称，以简化消息发送。要求唯一行为的子类将包含父类中列出的同样的行为。当兼职雇员收到“计算报酬”消息时，它将自动使用它自己的行为列表中的计算报酬行为，因为它重载了从父类继承的行为。当改进一个现有系统时，多态性很有用，因为增加新类到现有的泛化/特化关系中以便满足新的业务规则或需求可能不可能或者不现实。



图 9-7 消息

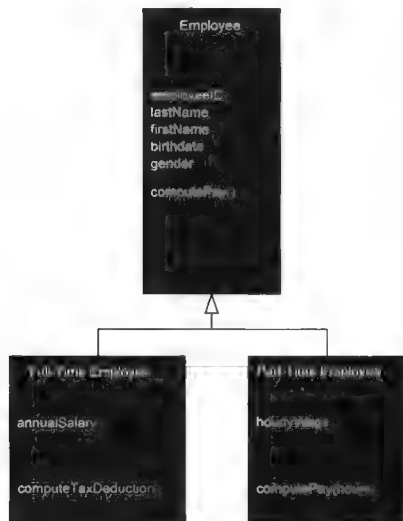


图 9-8 重载行为

那么多态性同消息发送有什么关系呢？请求服务的对象知道要请求什么服务（或行为）和从哪个对象请求，但是请求对象不需要关心行为如何实现。

9.3 UML 模型图

UML 模型图很像用于构造房屋的蓝图，不过蓝图一般是提供给建筑工人用来铺设管道、安装电路以及取暖和空调设备等用的，而每个 UML 模型图则为开发团队提供从不同视角看到的信息系统。

图 9-9 描述了 UML 2.0 的 13 种模型图。这个清单不是按照字母顺序组织的，而是为了让每个图形的描述都能基于它上面图形的描述。需要一整门大学课程来深入学习这些模型图。当我们研究系统分析生命周期的概貌时，本书有三个章节将深入研究核心的 UML 模型图。

第6章——需求分析阶段

- 用例图

第9章——逻辑设计阶段

- 活动图
- 系统顺序图（一种高层顺序图）
- 类图

第17章——物理设计阶段

- 顺序图
- 类图（含有更多细节）
- 状态机图
- 通信图
- 组件图
- 部署图

模型图	描 述
用例图	描述系统与外部系统和用户的交互。换句话说，它们以图形化的方式描述了谁将使用系统，以及用户期望以什么方式与系统交互。用例描述也用于以文本化的方式描述每个交互步骤的顺序。
活动图	描述一个业务过程或者一个用例的活动的顺序流。它也可以用于系统的建模逻辑。
类图	描述系统的对象结构，它们显示构成系统的对象类，以及那些对象类之间的关系。
对象图	类似于类图，但并不描述对象类，它们对实际的对象实例进行建模——显示实例的属性的当前值。对象图为开发人员提供对象在某个时间点上的“快照”。
状态机图	用于建模在生命周期中事件如何改变对象的状态——对象可以经历的各种状态，以及引起对象从一个状态向另一个状态转换的事件。
组合结构图	分解类、组件或用例的内部结构。
顺序图	以图形化的方式描述了在一个用例或操作的执行过程中对象如何通过消息互相交互，说明了消息如何在对象之间被发送和接收以及发送的顺序。
通信图	在 UML 1. X 中称为协作图，描述了对对象通过消息的交互。因此，它类似于顺序图。但顺序图关注消息的定时或“顺序”，通信图则以一种网络格式表现对象之间的结构化组织。
交互图	组合了顺序图和活动图的特征，显示在每个用例的活动中对象如何交互。
定时图	另一种交互图，它关注一个对象或一组对象在改变状态时的时间约束条件。当为设备设计嵌入式软件时，定时图特别有用。
组件图	描述程序代码分解成组件的组织结构，以及组件之间的交互。
部署图	描述软件组件在系统的硬件“节点”的物理体系结构中的配置。
包图	描述类或其他 UML 构件如何组织成包（对应 Java 的包或者 C++ 和 .NET 的名字空间），以及这些包之间的依赖关系。

图 9-9 UML 2.0 模型图

9.4 对象建模过程

像任何其他系统分析方法一样，面向对象分析（OOA）的目的是更好地理解系统及其功能需求。换句话说，OOA 要求我们辨识从用户角度开发所需的系统功能、支持所需系统功能的对象、对象的数据属性、相关的行为以及对象之间的关联。在第 6 章中，我们介绍了用例模型，用于确定所需的系统功能。在本章中，我们将学习细化第 6 章创建的用例模型，通过对象建模记录确定的对象、对象封装的数据和行为以及对象之间的关系。

面向对象分析包含 3 个活动：

- 1. 建模系统功能。
- 2. 发现并确定业务对象。
- 3. 组织对象并确定其关系。

9.4.1 建模系统的功能性描述

在第 6 章中，我们介绍了用例建模过程，它使用业务需求用例建模系统功能需求。在此期间，记录的用例仅仅包含了有关业务事件的一般性信息，其目标是快速地记录所有的业务事件（用例），以便定义和验证需求。在进行面向对象分析时，每个以前定义的用例将基于开发过程中了解到的事实被细化，以包括更多的细节，例如用户界面需求。为了准备进行对象建模，我们需要将业务需求用例模型转换成分析用例模型。

9.4.2 构造分析用例模型

在面向对象分析中，我们通过以下步骤将业务需求用例模型转换成分析用例模型：

- 1. 确定、定义并记录新的参与者。
- 2. 确定、定义并记录新的用例。
- 3. 确定任何复用的可能性。
- 4. 细化用例模型图（如果需要）。
- 5. 记录系统分析用例描述。

9.4.2.1 第1步：确定、定义并记录新的参与者

在创建了业务需求用例模型和其最终被系统所有者批准之间，系统分析员和开发团队的其他成员通过与关联人员交谈和研究项目发布物，继续了解系统成功还需要什么。通过这些努力，有可能会发现需要被定义和记录的额外参与者。例如，当分析俱乐部会员发起的下新订单用例（见图6-13）时，我们确定俱乐部会员要能够通过因特网输入订单信息，但会员也可以通过邮件提交订单。为了使订单信息能够输入到系统中，某个人要同系统交互实现此功能，因此需要另一个参与者。新确定的参与者称为会员服务代理，以及其他新参与者，都需要在前面准备好的参与者词汇表中定义（见图6-8）。

9.4.2.2 第2步：确定、定义并记录新的用例

（第1步发现的）新的参与者“会员服务代理”导致了与系统的一个新的交互——新的用例。作为一般规则，用于处理业务事件的每种类型的用户接口都需要自己的用例。以银行业为例，在ATM机中存款的用例将不同于在柜台存款用例。处理的目相同，许多步骤也相同，但实际的系统用户可能不同，用户采用特定的技术（ATM机以及为银行出纳设计的GUI界面）如何与系统交互可能不同。新确定的用例需要被定义到前面准备好的用例参与者词汇表中（见图6-10）。

9.4.2.3 第3步：确定任何复用的可能性

如上面第2步所述，当两个用例有同样的业务目标但接口技术和实际的系统用户不同时，两个用例可能共享公共的步骤。回顾第6章，为了消除冗余步骤，我们可以将这些公共步骤提取成独立的用例，称为抽象用例。而且，当我们分析用例并发现某个用例包含多个步骤构成的复杂功能（使其难以理解）时，可以将更复杂的用例提取成专门的用例，称为扩展用例。新的用例也需要被定义到前面准备好的用例参与者词汇表中。

9.4.2.4 第4步：细化用例模型图（如果需要）

由于发现了新的参与者和/或用例，我们现在需要修改前面构造的用例模型图（图6-10）以包括进这些内容。图9-10是修订后的用例模型图，它包含了新确定的参与者“会员服务代理”、新确定的用例“输入新会员订单”和“确定合适的分销中心并分发填写的订单”。

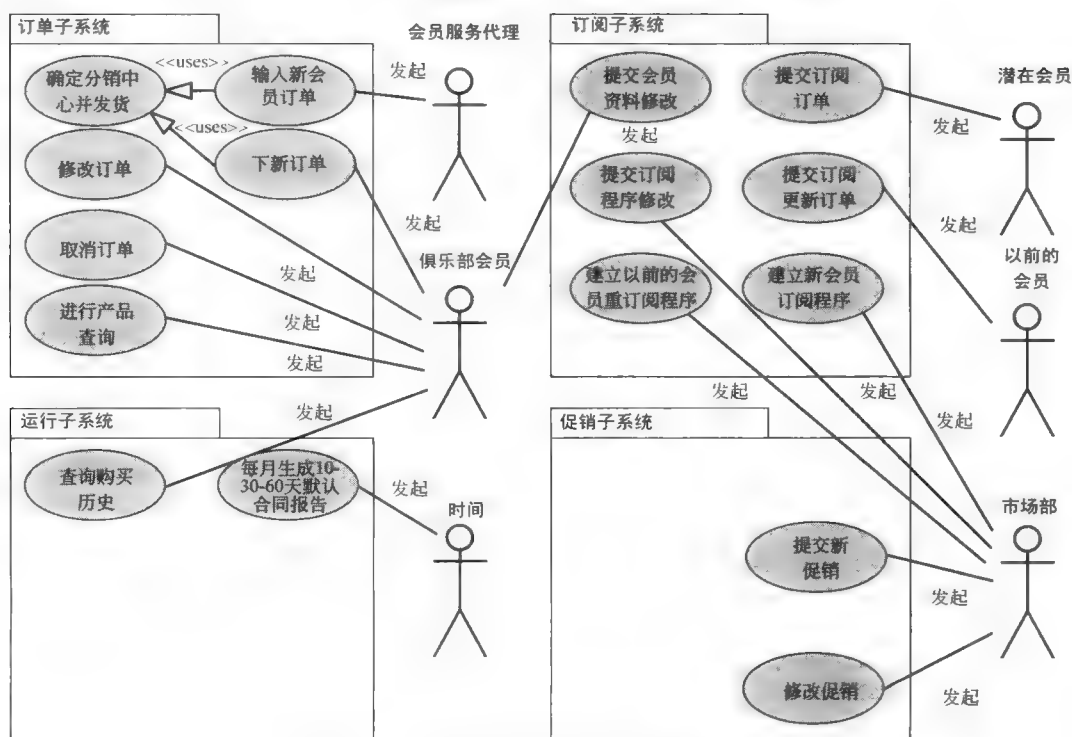


图9-10 修订后的会员服务系统用例模型图

9.4.2.5 第5步：记录系统分析用例描述

一旦修订了所有的业务需求用例并被用户批准，将细化每个用例包含更多的信息以便详细地说明系统功能。得到的用例称为系统分析用例，应该没有太多的实现细节，除了描述系统用户用来与系统交互的手段（Windows GUI、Internet 浏览器、电话等）的高层信息。系统分析用例包含从系统用户角度的描述，比业务需求用例（与系统）交互得更自然。图 9-11 和图 9-12 是业务需求用例“下新订单”的细化。图 9-11 描述了“俱乐部会员”作为主要系统参与者，使用系统输入订单；图 9-12 描述了“会员服务代理”使用系统输入从俱乐部会员收到的订单信息。

会员服务系统

作者：K. Dittman

日期：11/01/06

版本：1.00

用例名称	下新订单	用例类型 业务需求： <input type="checkbox"/> 系统分析： <input checked="" type="checkbox"/> ●
用例 ID	MSS-SUC002. 00	
优先权	高	
来源	需求——MSS-R1. 00 需求用例——MSS-BUC002. 00	
主要业务参与者	俱乐部会员（别名——活动会员、会员）	
主要系统参与者	俱乐部会员（别名——活动会员、会员） ●	
其他参与者	● <ul style="list-style-type: none">仓库（别名——分销中心）（外部接收者）应付账/应收账（外部服务者）	
其他有兴趣的关联人员	● <ul style="list-style-type: none">市场部——对销售活动感兴趣，为了计划新的促销采购部——对销售活动感兴趣，为了补充库存管理层——对销售活动感兴趣，为了评估公司性能和顾客满意度	
描述	该用例描述一个俱乐部会员通过因特网提交一个音阶娱乐俱乐部产品的订单。会员选择他想购买的项目。一旦会员完成了采购，会员的资料信息以及他的账号被验证。一旦验证产品有库存，就向仓库发出一个发货订单准备发货。对于没有库存的产品，生成一个退单。一旦完成，会员将得到一份订单证实。	
前置条件	提交订单的一方（个人或者公司）必须是活动的俱乐部会员。 会员必须登录到系统中（提供身份认证）输入订单。	
触发器	当会员选择输入新订单时，用例被触发。	
典型事件过程	参与者动作	系统响应
	<p>第 1 步：会员请求输入新订单。</p> <p>第 3 步：会员浏览可得到的条目，选择他想购买的项及数量。</p> <p>第 5 步：会员验证个人信息（发货和收费地址）。如果没有变化，会员相应地响应（继续）。</p> <p>第 7 步：会员验证订单。如果没有变化，会员相应地响应（继续）。</p> <p>第 9 步：会员选择期望的支付方式。</p> <p>第 11 步：会员验证订单。如果没有变化，会员相应地响应（继续）。</p>	<p>第 2 步：系统做出响应，显示产品目录。</p> <p>第 4 步：一旦会员完成了选择，系统访问文件，显示会员的个人信息（发货和收费地址）。</p> <p>第 6 步：对于订购的每个产品，系统验证产品可用性，决定发货日期，决定向俱乐部会员收取的价格，决定订单的总价格。如果某项不能马上得到，指出产品退单，或者还没有发货（对于预定）。如果某项不再可得到，也需要指出。系统然后给会员显示一个订单总结供确认。</p> <p>第 8 步：系统检查俱乐部会员账号的状态。如果满足，系统提示会员选择期望的支付方式（以后支付还是使用信用卡立即支付）。</p> <p>第 10 步：系统显示订单总结，包括期望的支付方式，供会员确认。</p> <p>第 12 步：系统记录订单信息（如果需要还包括退单）。</p> <p>第 13 步：调用抽象用例 MSS-AUC001. 00（确定合适的分销中心并分发填写的订单）。</p> <p>第 14 步：一旦订单被处理，系统生成一个订单确认，把它显示给会员，并通过电子邮件发送给会员。</p>

图 9-11 下新订单用例

替代事件过程	<p>替代第3步：会员输入查询条件获取特定的项目，或者显示一个缩减的列表以便于浏览和购买。</p> <p>替代第5步：如果需要修改，会员修改相应的发货地址、收费地址或电子邮件，并告诉系统相应地存储。系统将验证修改，如果成功，将把新信息存储到文件中。</p> <p>替代第7步：如果订单要修改，会员可以删除不再想要的项目，或者修改订购数量。一旦会员完成了订单修改，系统继续处理订单（转到第6步）。如果会员请求继续购买，转到第3步。如果会员需要修改个人信息，转到第5步。</p> <p>替代第8步：如果俱乐部会员的账号状态不良，给会员显示账号状态，订单被挂起的原因，以及解决问题所需的动作。另外，同样信息的电子邮件发送给会员。系统提示会员挂起订单日后处理或者取消订单。如果会员希望挂起订单，系统记录订单信息并把它设成挂起状态，然后显示音阶主页。如果会员选择取消订单，系统清除输入的信息，然后显示音阶主页。终止用例。</p> <p>替代第10步：如果会员选择通过信用卡支付，系统提示会员输入信用卡信息（卡号和有效期），提醒用户文件中的付款地址必须与信用卡提供的付款地址一致。会员输入所需的信息，然后请求系统继续。系统验证提供的信用卡账号。如果无法验证账号，系统通知会员，并请求其他支付方式。如果会员不能同时提供另一种支付方式，他可以选择挂起或者取消订单。如果会员希望挂起订单，系统记录订单信息，并把它设置成挂起状态，然后显示音阶主页。如果会员选择取消订单，系统清除输入的信息，然后显示音阶主页。终止用例。</p>
替代事件过程	<p>替代第11步：如果要修改订单，会员可以删除不再想要的项目，或者修改订购数量。一旦会员完成了订单修改，系统继续处理订单（转到第6步）。如果会员请求继续购买，转到第3步。如果会员需要修改个人信息，转到第5步。</p> <p>替代第12步：如果所有的订购项目都被退单，订单就不发送到分销中心。</p>
结论	当俱乐部会员收到订单确认时，该用例结束。
后置条件	订单被记录下来，如果订购的产品有货，将发货。对于缺货的产品，生成一个延迟交货单。
业务规则	<ul style="list-style-type: none"> • 会员必须拥有一个有效的电子邮件地址用于提交联机订单。 • 只有当产品发货时，才向俱乐部会员收费。
实现约束和说明	<ul style="list-style-type: none"> • 用例必须对会员 24×7 可用。 • 频率——估计用例每天执行 3500 次，应支持最多 50 个并发会员。
假设	<ul style="list-style-type: none"> • 产品能够通过分销中心发送以履行订单。 • 将在日报表中通知采购部门延迟交货单（独立的用例）。 • 会员响应促销或者使用信用卡可能会影响每个订购项目的价格。 • 会员可以在任何时候取消订单。
开放问题	无

图 9-11 （续）

系统分析用例在生命周期的设计阶段将被进一步细化，以说明如何实现。在进入设计阶段之前所有开放的问题和要确定（TBD）的问题都被解决了，这一点很重要，因为每个决策都会影响设计的特性。请注意在系统分析用例中的以下部分。

- ① 用例类型——在进行用例建模时，构造的第1个用例是业务需求用例，它关注各种关联人员的战略视图和目标。这类用例是面向业务的，反映了系统期望行为的高层视图。其中没有技术细节，并可以包含手工活动和将被自动化的活动。业务需求用例提供了对问题领域和范围的一般性理解，但它们并不包含与开发人员交流系统应如何操作所需的细节。

为了反映用户界面约束之类的实现细节，从业务用例中导出战术性的用例，称为系统用例。可以从一个业务用例中导出一个或多个系统分析用例。开发人员使用这种用例说明详细的需求，辅助估计和规划，交流编程需求，形成用户文档的基础。每个系统用例对应一个测试用例，用于验证系统满足客户的需求。

在整个系统开发生命周期中，系统用例继续细化。当遵循迭代开发方法时，按照从需求到分析到设计跟踪每个用例处于什么位置是明智的。

会员服务系统

作者: K. Dittman

日期: 11/01/06

版本: 1.00

用例名称	输入新会员订单		用例类型 业务需求: <input type="checkbox"/> 系统分析: <input checked="" type="checkbox"/>
用例 ID	MSS-SUC003.00		
优先权	高		
来源	需求——MSS-R1.00 需求用例——MSS-BUC002.00		
主要业务参与者	俱乐部会员（别名——活动会员、会员）		
主要系统参与者	会员服务代理（别名——用户）		
其他参与者	• 仓库（别名——分销中心）（外部接收者） • 应付账/应收账（外部服务者）		
其他有兴趣的关联人员	• 市场部——对销售活动感兴趣，为了计划新的促销 • 采购部——对销售活动感兴趣，为了补充库存 • 管理层——对销售活动感兴趣，为了评估公司性能和顾客满意度		
描述	该用例描述一个会员服务代理输入产品的新订单的事件，订单或者由会员通过电子邮件提交，或者由电话提交。会员的资料信息以及他的账号被验证。一旦验证产品有库存，就向仓库发出一个发货订单准备发货。对于没有库存的产品，生成一个退单。一旦完成，会员将得到一份订单证实。		
前置条件	提交订单的人必须是活动的俱乐部会员。 会员服务代理必须登录到系统中。		
触发器	当会员服务代理选择输入新订单时，用例被触发		
典型事件过程	参与者动作	系统响应	
	<div>第 1 步：会员服务代理请求输入新订单。</div> <div>第 3 步：会员服务代理提供会员姓名或 ID。</div> <div>第 5 步：会员服务代理验证个人信息（发货和收费地址）。如果没有变化，会员服务代理相应地响应（继续）。</div> <div>第 7 步：会员服务代理输入订购的每项产品的 ID 和数量。</div> <div>第 10 步：会员服务代理根据会员提供的信息验证订单。如果没有变化，会员相应地响应（继续）。</div> <div>第 12 步：会员服务代理选择会员指定的期望的支付方式。</div> <div>第 14 步：会员服务代理验证订单。如果没有变化，会员相应地响应（继续）。</div> <div>第 2 步：系统做出响应，提示用户输入提交订单的会员的 ID 或姓名。</div> <div>第 4 步：系统访问文件中会员的个人信息并显示给用户。如果存在多个用户匹配用户提供的信息，系统显示一份清单，并提示用户选择正确的那个。</div> <div>第 6 步：系统做出响应，提示用户输入订购的每项产品的 ID 和数量。</div> <div>第 8 步：对于订购的每个产品，系统验证产品标识。</div> <div>第 9 步：对于订购的每个产品，系统验证产品可用性，决定发货日期，决定向俱乐部会员收取的价格，决定订单的总价格。如果某项不能马上得到，指出产品退单，或者还没有发货（对于预订）。如果某项不再可得到，也需要指出。系统然后给会员显示一个订单总结供确认。</div> <div>第 11 步：系统检查俱乐部会员账号的状态。如果满足，系统提示会员选择期望的支付方式（以后支付还是使用信用卡立即支付）。</div> <div>第 13 步：系统显示订单的最终总结，包括期望的支付方式，供用户确认。</div> <div>第 15 步：系统记录订单信息（如果需要还包括退单）。</div> <div>第 16 步：调用抽象用例 MSS-AUC001.00（确定合适的分销中心并分发填写的订单）。</div> <div>第 17 步：一旦订单被处理，系统生成一个订单确认，把它显示给会员服务代理，并通过电子邮件发送给会员。</div>		

图 9-12 输入新会员订单用例

替代事件过程	<p>替代第4步：如果文件中找不到会员，通知用户存在差异。</p> <p>替代第5步：如果需要修改，会员修改相应的发货地址、收费地址或电子邮件，并告诉系统相应地存储。系统将验证修改，如果成功，将把新信息存储到文件中。</p> <p>替代第8步：如果提供的产品信息无法匹配任何产品，系统给用户显示差异，并提示用户澄清。</p> <p>替代第4步：如果订单要修改，用户可以删除不再想要的项目，或者修改订购数量。一旦会员完成了订单修改，系统继续处理订单（转到第8步）。</p> <p>替代第11步：如果俱乐部会员的账号状态不良，给用户显示账号状态，订单被挂起的原因，以及解决问题所需的动作。（会员服务代理可能需要日后联系会员来解决；另外，如果会员拥有有效的电子邮件地址，则一份同样信息的电子邮件将发送给会员。）系统提示用户挂起订单日后处理或者取消订单。如果用户希望挂起订单，系统记录订单信息并把它设成挂起状态，然后显示音阶主页。如果用户选择取消订单，系统清除输入的信息，然后显示音阶主页。终止用例。</p> <p>替代第13步：如果用户选择通过信用卡支付，系统提示用户输入会员提供的信用卡信息（卡号和有效期），提醒用户文件中的付款地址必须与信用卡提供的付款地址一致。用户输入所需的信息，然后请求系统继续。系统验证提供的信用卡账号。如果无法验证账号，系统通知用户，并请求其他支付方式。如果用户不能同时提供另一种支付方式，他可以选择挂起或者取消订单。如果用户希望挂起订单，系统记录订单信息，并把它设置成挂起状态，然后显示音阶主页。如果用户选择取消订单，系统清除输入的信息，然后显示音阶主页。终止用例。</p> <p>替代第14步：如果要修改订单，用户可以删除不再想要的项目，或者修改订购数量。一旦会员完成了订单修改，系统继续处理订单（转到第8步）。</p> <p>替代第15步：如果所有的订购项目都被退单，订单就不发送到分销中心。</p>
结论	当俱乐部会员收到订单确认时，该用例结束。
后续条件	订单被记录下来，如果订购的产品有货，将发货。对于缺货的产品，生成一个延迟交货单。
业务规则	<ul style="list-style-type: none"> • 会员必须拥有一个有效的电子邮件地址用于提交联机订单。 • 只有当产品发货时，才向俱乐部会员收费。
实现约束和说明	<ul style="list-style-type: none"> • 用例必须对会员服务代理东部时间上午7点到下午9点可用。 • 频率——估计用例每天执行4500次，应支持最多25个并发用户。
假设	<ul style="list-style-type: none"> • 产品能够通过分销中心发送以履行订单。 • 将在日报中通知采购部门延迟交货单（独立的用例）。 • 会员响应促销或者使用信用卡可能会影响每个订购项目的价格。 • 会员可以在任何时候取消订单。
开放问题	无

图9-12 （续）

②主要系统参与者——主要系统参与者是实际使用系统和与系统打交道的关联人员。用户界面就是为这些关联人员设计的。

③抽象用例——调用一个抽象用例的例子。

9.4.2.6 记录抽象用例描述和扩展用例描述

记录扩展用例和抽象用例的描述与记录常规用例的描述十分类似，但存在几个主要的区别。抽象和扩展用例不是由参与者发起，而是由其他用例调用的。而且，抽象和扩展用例一般比较短，不需要那么多数据域。图9-13是抽象用例的例子。请注意描述内容的差别。

①用例类型——当被两个以上用例调用时，采用抽象用例。当扩展单个用例的功能时，采用扩展用例。

②调用者——调用特定用例的用例的ID和名字。

注意 抽象用例可以调用其他抽象和/或扩展用例，扩展用例也可以调用其他抽象和/或扩展用例——因而提供了许多用例复用。

当完成系统分析用例的定义之后，现在就可以开始确定用例中包含的对象了。这些对象表示了业务领域的事物或实体——我们对其感兴趣，而且想收集有关信息。在这里，将用一两句话集中描述这些对象，以后会扩展我们的定义，以包含了解到的关于每个对象的更多的细节信息。

会员服务系统

作者: K. Dittman

日期: 11/01/06

版本: 1.00

用例名称	确定合适的分销中心并分发填写的订单	用例类型 抽象: <input checked="" type="checkbox"/> 扩展: <input type="checkbox"/> ①
用例 ID	MSS-AUC001.00	
优先权	高	
来源	MSS-SUC002.00 MSS-SUC003.00	
参与者	• 仓库（别名——分销中心）（外部接收者）	
描述	该用例描述选择分销中心事件，该分销中心服务俱乐部会员为某个特定订单提供的发货地址。订单信息（发货单）被发送到（分发）该分销中心履行。	
前置条件	订单准备好分发到合适的分销中心。	
典型事件过程	第 1 步：系统基于发货地址的州和邮编选择恰当的分销中心。 第 2 步：一旦选择了分销中心，就形成一份包含发货项目的发货单。 第 3 步：发货单传输到分销中心（发货和收货系统），用于准备发货。	
替代事件过程	替代第 1 步：如果发货地址是国际地址，将发货单发送到印第安纳州。	
后置条件	发货单传输到合适的分销中心。	

图 9-13 抽象用例举例

9.4.3 建模用例活动

UML 提供了一种活动图用于建模系统的过程步骤或活动。它们类似于流程图, 图形化地描述了业务过程或用例的活动的顺序流程。它们不同于流程图, 它们提供了描述并行活动的机制。因此, 它们特别适用于建模这样的活动——当操作正在执行时的活动, 以及那些活动的结果, 例如建模引起窗口显示和关闭的事件。活动图很灵活, 既可以用于分析阶段, 也可以用于设计阶段。图 9-14 是根据用例“输入新会员订单”构造的活动图。对于每个用例, 至少可以构造一个活动图。如果用例很长, 包含复杂的逻辑, 则可以构造多个活动图。系统分析员使用活动图可以更好地理解用例步骤的流程和顺序。

图 9-14 展示了以下活动图记号:

- ① 初始节点——实心圆表示过程的开始。
- ② 动作——圆角矩形表示单个步骤。动作的序列构成了图形描述的活动。
- ③ 流——图上的箭头指示通过动作的进展。除了从决策中流出, 大多数流不需要文字说明。
- ④ 决策——具有一个进入流和两个或多个输出流的菱形图形。输出流被标记以指示条件。
- ⑤ 合并——具有两个或多个进入流和一个输出流的菱形图形。它将以前被决策分开的流组合在一起。任何一个流到达合并, 过程就继续执行。
- ⑥ 分支——具有一个进入流和两个或多个输出流的黑条。分支下面的并行流中的动作可以按照任何顺序出现或者并发。
- ⑦ 联合——具有两个或多个进入流和一个输出流的黑条, 表示并发处理的结束。在过程继续执行之前, 所有的进入联合的流都必须完成。
- ⑧ 活动终止——内部空心的实心圆表示过程的结束。

图 9-14 中展示的活动图以图形化方式描述了用例的步骤, 但它没有说明谁在执行这些步骤。这可能不是个问题。经常你绘制活动图只是进行逻辑上的处理。但如果你想说明谁做什么, 你可以将活动图分解成展示一个特定类或角色执行的工作的几块。图 9-15 是“下新订单”用例 (见图 9-11) 的活动图, 按照会员和系统的动作进行了一个简单的一维分割。分割的部分经常被称为泳道, 因为它们就像游泳运动员使用的赛道。一个活动图可以具有三个或更多的泳道以显示接收者角色。还可以将活动图分割成二维的栅格。

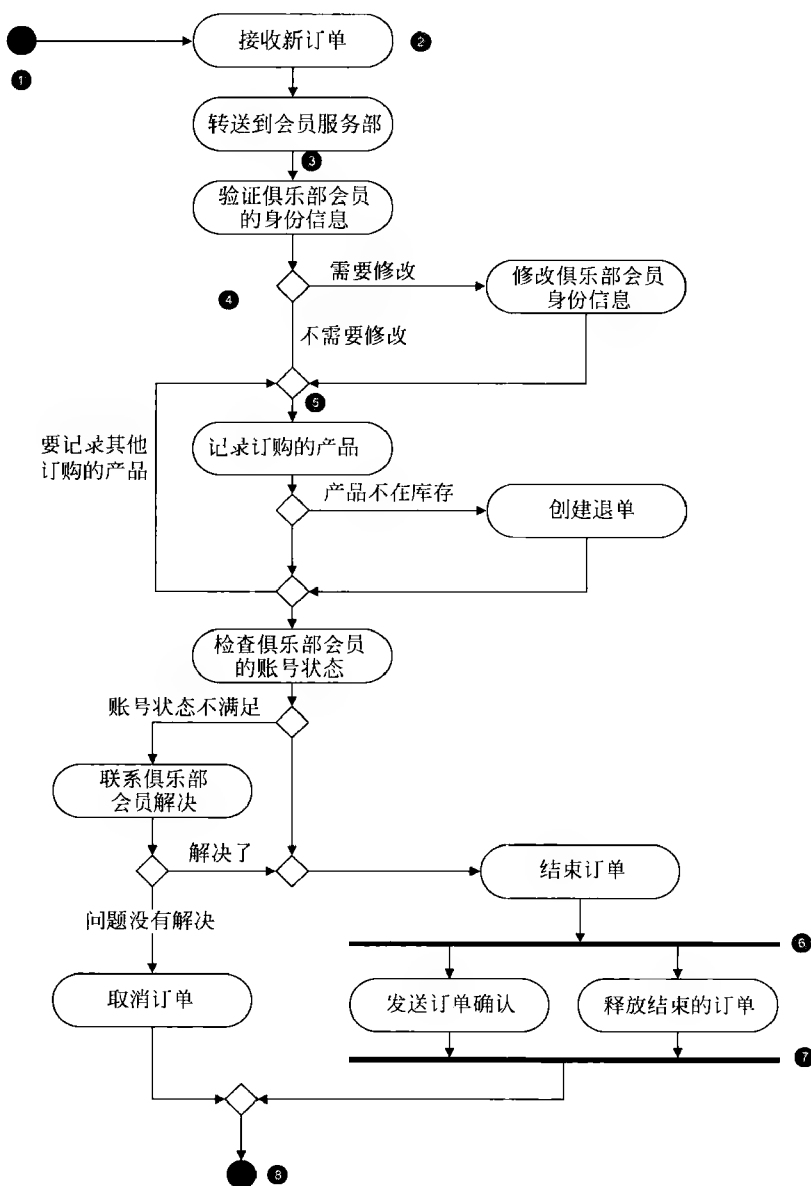


图 9-14 “输入新会员订单”用例的活动图

图 9-15 展示了活动图的另外两个特征：

- ❶ 子活动指示器——在动作中的靶子符号指示这个动作被分解成另一个独立的活动图。这有助于你防止活动图变得过度复杂。
- ❷ 连接器——在一个圆圈内的字母提供了另一种管理复杂度的工具。进入连接器的流跳转到具有相匹配字母的连接器的输出流。

这两个例子并没有用尽活动图的所有功能。动作可以被基于时间或者某个外部过程的信号调用。你甚至可以说明参数传递以及其他特殊种类的信息。但我们已经涉及了足够的内容，你可以开始绘制活动图了。

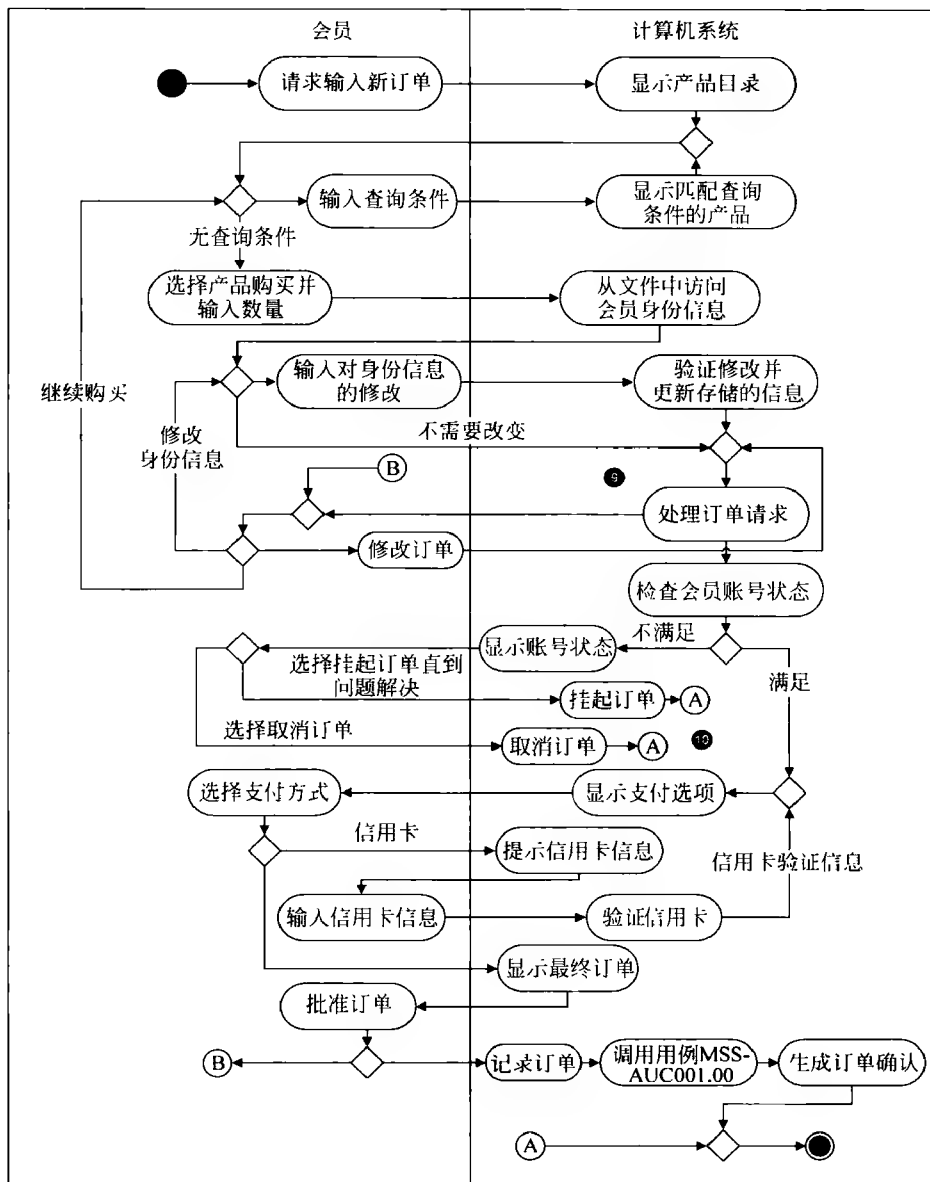


图 9-15 具有分割的“下新订单”用例的活动图

9.4.4 构造活动图指南

下面的清单表示一个较好的构造活动图的过程：

- 从一个作为起点的初始节点开始。
- 如果它们与你的分析有关就增加分割。
- 为用例的每个主要步骤（或者一个角色发起的每个主要步骤）添加一个动作。
- 从一个活动到另一个活动、决策点或者终点添加一条流。为了含义的最大精确度，每个动作应该只有一个输入流和一个输出流，所有的分支、联合、决策和合并例外。
- 在流分解成不同的路线的地方添加决策。确保用一个合并将各个流重新合并。
- 在并行执行活动的地方添加分支和联合。
- 用一个单一的活动终止符号结束。

9.4.5 绘制系统顺序图

在逻辑设计阶段一些OO方法使用的另一种工具是系统顺序图（system sequence diagram）。如前所述，顺序图描述了在用例执行或操作过程中对象如何通过消息相互交互。我们还没有开始分析单个对象类；将在后面介绍，当我们构造了第一个版本的类图时。现在，我们仍把系统作为一个整体来考虑。

正如我们所说的，面向对象的世界是由对象之间发送的消息驱动的。系统顺序图帮助我们开始确定进入和退出系统的高层消息。以后这些消息将成为单个对象的责任，对象将通过与其他对象交互完成这些责任。将在第17章再讲述。

图9-16显示了“提交新订单”用例的一幅系统顺序图。注意系统顺序图没有包括用例任何的可选过程。它通过用例的单一一条路径描述了单个场景。所以对于一个用例来说，一套完整的系统顺序图可能包括几幅图形。

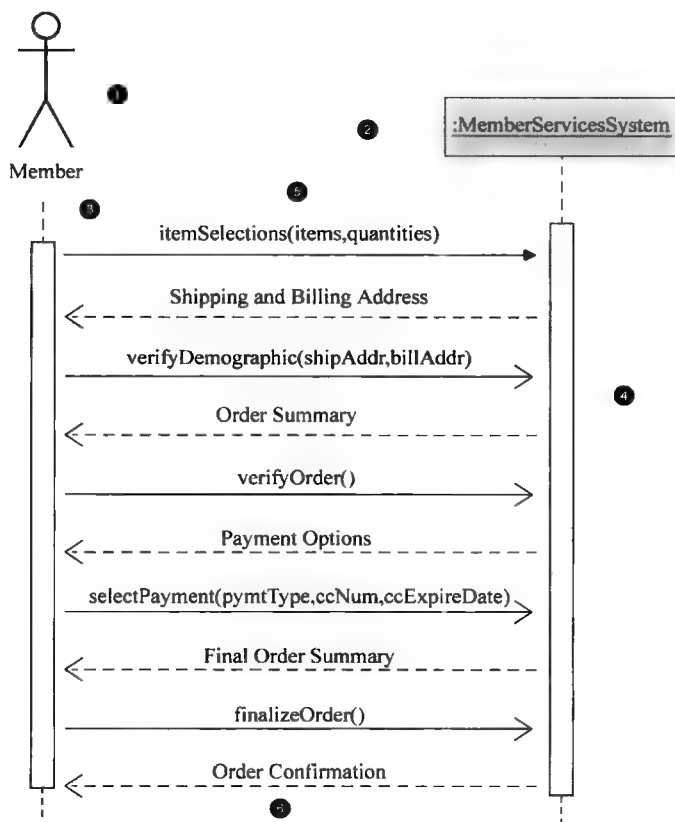


图9-16 “提交新订单”用例的系统顺序图

图9-16展示了以下系统顺序图的记号：

- ①角色——用例的发起角色使用用例角色符号表示。
- ②系统——盒子表示系统作为一个“黑盒子”或者作为一个整体。冒号（:）是标准的顺序图记号用来表示系统的一个运行“实例”。
- ③生命线——从角色和系统符号向下延伸的垂直虚线，表示生命顺序。
- ④活动条——放置在生命线的条形表示参与者进行交互活动的一段时间。有些方法学在系统顺序图中不包含它们，但我们把它们包含进来以同完全的顺序图保持一致。
- ⑤输入消息——从角色到系统的水平箭头表示消息输入。UML的消息命名规范是用开始的第一个单词字母小写，后续的单词起始字母大写，单词之间没有空格。括号内包含了你在此时知道的任何参数，按照同样的命名规范，并用逗号分隔每个参数。你可能想知道用户如何传递这些消

息。用户将同用户界面交互，用户界面通过合适的格式为用户传递消息。我们将在第 17 章更详细地讲述。

- ④ 输出消息——从系统到角色的显示成虚线的水平箭头。由于这些消息采用 Web 表单、报告、电子邮件等形式，所以这些消息不需要使用标准的命名规范，如果你想用也可以用。

图 9-17 是注册验证的系统顺序图，它展示了以下其他的记号：

- ⑤ 接收者角色——也可以包括从系统接收消息的其他角色或外部代理。
- ⑥ 框体——一个可以封装一条或多条消息的盒子，用来分割一段顺序图。它们可以用来表示循环、可选片断或者可选步骤。对于一个可选片断，方括号中显示的条件表示在该条件下会执行步骤。

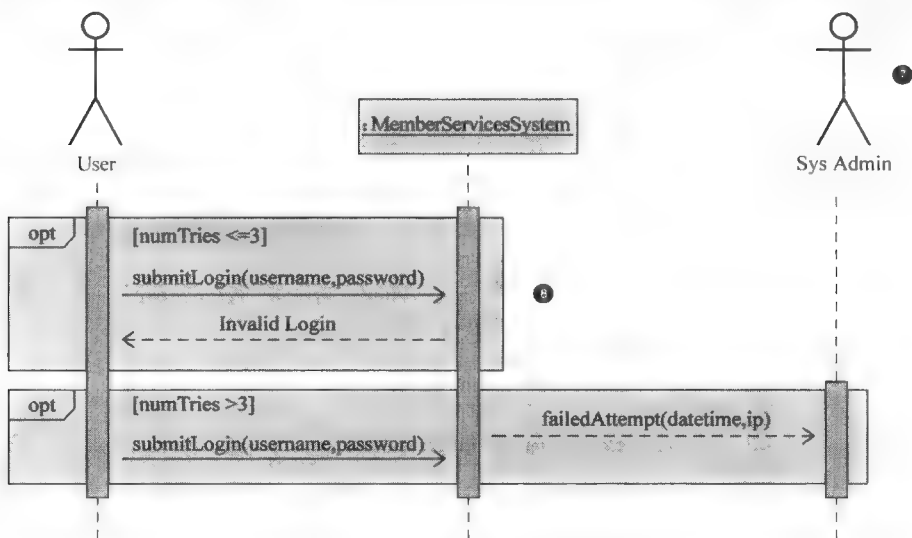


图 9-17 注册验证的系统顺序图

9.4.6 构造系统顺序图指南

- 确定你将描述用例的哪个场景。系统顺序图的目的是发现消息，而不是建模逻辑。所以尽管你可以包含整个用例的可选消息，但清楚地描述单个场景更重要。
- 绘制一个矩形表示作为一个整体的系统，并在其下面延伸生命线。
- 确定每个直接给系统提供一条输入的角色，或者直接从系统接收输出的角色。在角色下延伸生命线。
- 检查用例描述来确定系统的输入和输出。忽略系统内部消息。每条外部消息从角色的生命线到系统的生命线绘制一条水平箭头线，或者从系统的生命线到角色的生命线绘制一条水平箭头线。按照 UML 命名规范标记输入，这将有助于确定业务对象的行为和属性。
- 添加框体以表示带条件的可选消息。框体也可以表示循环和替代片断（这些将在第 17 章讨论）。
- 自顶向下验证消息按照正确的顺序显示出来。

9.4.7 发现和确定业务对象

在试图确定对象时，许多方法学专家建议查找需求文档或者其他相关文档，并标记出表示潜在对象的名词。这可能是一项复杂的任务。因为名词太多了。用例建模为这个问题提供了一个解决方案，即将整个系统分解成用例。这些相对较小的用例减少了挑选名词的工作量，并且使该技术更为有效。下面介绍在用例建模中用来确定和发现业务对象的步骤。

9.4.7.1 第 1 步：发现潜在对象

这一步需要检查每个用例以发现对应业务实体或事件的名词。例如，图 9-18 描述了“下新订单”用例，其中所有名词都标了出来。在用例中发现的每个名词都被添加到潜在对象列表中，供进一步分析之用（见图 9-19）。

9.4.7.2 第2步：筛选建议的对象

不是在列表中的所有候选（名词）都表示我们的问题域内的有用的业务对象。通过分析每个候选，并询问以下问题，我们能够决定候选是否应该保留，还是应该从列表中删除。

- 是否是另一个对象的同义词？
- 是否是系统范围之外的名词？
- 是否表示不具有独特行为角色的名词，或者表示外部角色的名词？
- 是否是进一步解释的不清楚的名词？
- 是否是描述了另一个对象的行动或属性的名词？

会员服务系统

作者：_____

日期：_____

版本：_____

用例名称	下新订单	用例类型 业务需求： <input type="checkbox"/> 系统分析： <input checked="" type="checkbox"/>
用例 ID	MSS-SUC002.00	
优先权	高	
来源	需求——MSS-R1.00 需求用例——MSS-BUC002.00	
主要业务参与者	俱乐部会员（别名——活动会员、会员）	
主要系统参与者	俱乐部会员（别名——活动会员、会员）	
其他参与者	<ul style="list-style-type: none">• 仓库（别名——分销中心）（外部接收者）• 应收账（外部服务者）	
其他有兴趣的关联人员	<ul style="list-style-type: none">• 市场部——对销售活动感兴趣，为了计划新的促销• 采购部——对销售活动感兴趣，为了补充库存• 管理层——对订购活动感兴趣，为了评估公司性能和顾客满意度	
描述	该用例描述一个俱乐部会员通过因特网提交一个音阶娱乐俱乐部产品的订单。会员选择他想购买的项目。一旦会员完成了采购，会员的资料信息以及他的账号被验证。一旦产品被验证有库存，就向仓库发出一个发货订单准备发货。对于没有库存的产品，生成一个退单。一旦完成，会员将得到一份订单证实。	
前置条件	提交订单的个人必须是活动的俱乐部会员。 会员必须登录到系统中（提供身份认证）输入订单。	
触发器	当会员选择输入新订单时，用例被触发。	
典型事件过程	参与者动作	系统响应
	<p>第1步：会员请求输入新订单。</p> <p>第3步：会员浏览可得到的条目，选择他想购买的项，以及数量。</p> <p>第5步：会员验证个人信息（发货和收费地址）。如果没有变化，会员相应地响应（继续）。</p> <p>第7步：会员验证订单。如果没有变化，会员相应地响应（继续）。</p> <p>第9步：会员选择期望的支付方式。</p> <p>第11步：会员验证订单。如果没有变化，会员相应地响应（继续）。</p>	<p>第2步：系统做出响应，显示产品目录。</p> <p>第4步：一旦会员完成了选择，系统访问文件，显示会员的个人信息（发货和收费地址）。</p> <p>第6步：对于订购的每个产品，系统验证产品可用性，决定发货日期，决定向俱乐部会员收取的价格，决定订单的总价格。如果某项不能马上得到，指出产品退单，或者还没有发货（对于预订）。如果某项不再可得到，也需要指出。系统然后给会员显示一个订单总结供确认。</p> <p>第8步：系统检查俱乐部会员账号的状态。如果满足，系统提示会员选择期望的支付方式（以后支付还是使用信用卡立即支付）。</p> <p>第10步：系统显示订单总结，包括期望的支付方式，供会员确认。</p> <p>第12步：系统记录订单信息（如果需要还包括退单）。</p> <p>第13步：调用抽象用例 MSS-AUC001.00（确定合适的分销中心并分发填写的订单）。</p> <p>第14步：一旦订单被处理，系统生成一个订单确认，把它显示给会员，并通过电子邮件发送给会员。</p>

图9-18 下新订单用例（名词用下划线标出）

替代事件过程	<p>替代第3步：会员输入查询条件获取特定的项目，或者显示一个缩减的列表以便于浏览和购买。</p> <p>替代第5步：如果需要修改，会员修改相应的发货地址、收费地址或电子邮件，并告诉系统相应地存储。系统将验证修改，如果成功，将把新信息存储到文件中。</p> <p>替代第7步：如果订单要修改，会员可以删除不再想要的项目，或者修改订购数量。一旦会员完成了订单修改，系统继续处理订单（转到第6步）。如果会员请求继续购买，转到第3步。如果会员需要修改个人信息，转到第5步。</p> <p>替代第8步：如果俱乐部会员的账号状态不良，给会员显示账号状态，订单被挂起的原因，以及解决问题所需的动作。另外，同样信息的电子邮件发送给会员。系统提示会员挂起订单日后处理或者取消订单。如果会员希望挂起订单，系统记录订单信息并把它设成挂起状态，然后显示音阶主页。如果会员选择取消订单，系统清除输入的信息，然后显示音阶主页。终止用例。</p> <p>替代第10步：如果会员选择通过信用卡支付，系统提示会员输入信用卡信息（卡号和有效期），提醒用户文件中的付款地址必须与信用卡提供的付款地址一致。会员输入所需的信息，然后请求系统继续。系统验证提供的信用卡账号。如果无法验证账号，系统通知会员，并请求其他支付方式。如果会员不能同时提供另一种支付方式，他可以选择挂起或者取消订单。如果会员希望挂起订单，系统记录订单信息，并把它设置成挂起状态，然后显示音阶主页。如果会员选择取消订单，系统清除输入的信息，然后显示音阶主页。终止用例。</p> <p>替代第11步：如果要修改订单，会员可以删除不再想要的项目，或者修改订购数量。一旦会员完成了订单修改，系统继续处理订单（转到第6步）。如果会员请求继续购买，转到第3步。如果会员需要修改个人信息，转到第5步。</p> <p>替代第12步：如果所有的订购项目都被退单，订单就不不发送到分销中心。</p>
结论	当俱乐部会员收到订单确认时，该用例结束。
后续条件	订单被记录下来，如果订购的产品有货，将发货。对于缺货的产品，生成一个延迟交货单。
业务规则	<ul style="list-style-type: none"> 会员必须拥有一个有效的电子邮件地址用于提交联机订单。 只有当产品发货时，才向俱乐部会员收费。
实现约束和说明	<ul style="list-style-type: none"> 用例必须对会员 24 × 7 可用。 频率——估计用例每天执行 3500 次，应支持最多 50 个并发会员。
假设	<ul style="list-style-type: none"> 产品能够通过分销中心发送以履行订单。 将在日报表中通知采购部门延迟交货单（独立的用例）。 会员响应促销或者使用信用卡可能会影响每个订购项目的价格。 会员可以在任何时候取消订单。
开放问题	无

图 9-18 （续）

如果对于某个候选名词，以上任何一个问题的答案都是“是”，那么候选应该从列表中删除。如果发现某个候选是属性，要把它们记录在另一个列表中，以防忘记。以后它们会用于构造类图。如果你对某个候选不确定，最好把这个候选保留在列表中，如果将来发现它不是对象而把它删除要比在类图构造好后再增加它要容易。

图 9-19 显示了“清理”候选对象列表的过程，“×”表示丢弃的候选对象，“√”表示保留的候选对象，表中还列出了对保留和丢弃候选对象的原因的解释。最后，图 9-20 表示了清理过程的结果，其中增加了在其他用例中已经发现的对象。

9.4.8 组织对象并确定其关系

一旦确定了系统的业务对象，就要组织这些对象，并记录对象之间的主要概念关系。类图以图形化的方式用来描述对象及其关联关系。在该图中还将包括多重性、关联关系、泛化/特化关系以及聚合关系。

9.4.8.1 第1步：确定关联关系和多重性

在这一步，需要确定存在于对象/类之间的关联关系。两个对象/类之间的关联关系是一个对象/类“需要知道”另一个对象/类的东西，关联关系允许一个对象/类交叉引用另一个对象/类，并能够向它发送消息。一旦确定了关联关系，我们就必须确定关联关系的多重性。

潜在对象	原因
Accounts Receivable	x 与当前项目无关
Actions	x 需要更好的关注——可能是 MEMBER ORDER 中的注释属性
Active Member	✓ MEMBER 的类型
Available Items	x PRODUCT 的同义词
Bank Order	x 采购系统的可响应的——与当前项目无关
Bank-Ordered	x 采购系统的可响应的——与当前项目无关
Billing Addresses	✓ ADDRESS 的类型
Catalog	x 同 PRODUCT，在面向对象设计中涉及的存在接口项目
Club Member	✓ MEMBER 的类型
Company Performance	x 与当前项目无关
Credit Card	✓ CREDIT CARD ACCOUNT
Credit Card Expiration Date	x CREDIT CARD ACCOUNT 的属性
Credit Card Number	x CREDIT CARD ACCOUNT 的属性
Creditor	x MEMBER 的属性
Customer Satisfaction	x 与当前项目无关
Delivery Items	x 在面向对象设计中涉及的存在接口项目
Demographic Information	x MEMBER 的属性
Distribution Center	✓ DISTRIBUTION CENTER
E-mail	x 在面向对象设计中涉及的存在接口项目
E-Mail Addresses	✓ ADDRESS 的类型
Event	x 与当前项目无关
Expended Ship Date	x MEMBER ORDERED PRODUCT 的属性
External Account	x 与当前项目无关
External Server	x 与当前项目无关
File	x 与当前项目无关
Gift Status	x MEMBER ORDER 的属性
Identification	x MEMBER 的属性
In Stock	x PRODUCT 的属性
Individual	x MEMBER 的同义词
Inventory	x PRODUCT 的属性
Items	x PRODUCT 的同义词
List	x 在面向对象设计中涉及的存在接口项目
Main Page	x 在面向对象设计中涉及的存在接口项目
Management	x 与当前项目无关
Marketing	x 与当前项目无关
Member	✓ MEMBER
Member Account Standing	x MEMBER 的属性
Member's Account Status	x MEMBER 的属性
New Order	✓ MEMBER ORDER
New Promotion	✓ PROMOTION
Options	x 在面向对象设计中涉及的存在接口项目
Order Activity	x 在面向对象设计中涉及的存在接口项目
Order Confirmation	x 在面向对象设计中涉及的存在接口项目
Order Total Cost	x MEMBER ORDER 的属性
Ordered Products	✓ MEMBER ORDERED PRODUCT
packing rules	x 在面向对象设计中涉及的存在接口项目
Payment Option	x MEMBER ORDER 的属性
Promoters	✓ MEMBER ORDER 的类型
Price to Be Charged	x MEMBER ORDERED PRODUCT 的属性
Position	x 需要更好的关注——可能是 MEMBER ORDER 中的注释属性
Procurement	x 与当前项目无关
Product Availability	x PRODUCT 的属性

图 9-19 分析潜在对象列表

潜在对象		原因
Product Ordered	x	MEMBER ORDERED PRODUCT 的同义词
Promotion	✓	PROMOTION
Purchase	x	MEMBER ORDER 的同义词
Quantity	x	MEMBER ORDERED PRODUCT 的属性
Reason	x	需要更好的关注——可能是 MEMBER ORDER 中的注释属性
Requirement	x	与目前项目无关
Requirements Use Case	x	与目前项目无关
Sales Activity	x	在面向对象设计中涉及的潜在接口项目
Search Criteria	x	在面向对象设计中涉及的潜在接口项目
Selections	x	MEMBER ORDERED PRODUCT 的同义词
Shipment	x	与目前项目无关——发货与接收的可响应性
Shipping Address	✓	ADDRESS 的类型
Shopping	x	在面向对象设计中涉及的潜在接口项目
SoundStage Products	x	PRODUCT 的同义词
Summary of the Order	x	在面向对象设计中涉及的潜在接口项目
System	x	与目前项目无关
Warehouse	x	DISTRIBUTION CENTER 的同义词
World wide Web	x	在面向对象设计中涉及的潜在接口项目

图 9-19 (续)

重要的是，分析员不只是一要确定明显的（或者由用户提供的）关联关系。确定关联关系的一种方式是使用对象/类矩阵，这个矩阵把对象/类作为列标题和行标题列出，然后可以检查矩阵中显示在一行上的每个对象/类与显示在一列上的每个对象/类之间可能的关联关系。关联关系的名称和多重性可以直接记录在矩阵的交叉单元格中。图 9-21 是一个包含了会员服务系统的建议的对象的例子矩阵。要解释单元格的内容，从左边的对象开始（行头），读出单元格内容，然后以列顶端的对象结束。例如：

- 一个“俱乐部会员”提交了零个或者多个“会员订单”。
- 一个“俱乐部会员”购买了零个或者多个“会员订购产品”。
- 一个“俱乐部会员”和“产品”之间没有关联关系。
- 一个“会员订单”由一个且仅一个“俱乐部会员”提交。
- 等等。

9.4.8.2 第2步：确定泛化/特化关系

一旦确定了基本的关联关系及其多重性，就必须确定是否存在泛化/特化关系。泛化/特化关系也称为分类层次或者“is a”关系，由超类（抽象类、父类）和子类（子类、具体类）构成。超类具有一般性，其中包含了层次中的公共属性和行为；子类则是特殊的，其中包含了那个对象专有的属性和行为，但它们也继承了超类的属性和行为。

泛化/特化关系可以通过类图查看。两个具有一对一多重性的对象之间存在关联关系吗？如果有，你能够说“对象 X 是对象 Y 的一种类型”吗？如果可以，我们就有了一个泛化/特化关系。另外，可以寻找具有公共属性和行为的对象，可以将公共属性和行为加以组合成为一个新的超类。为什么我们需要泛化/特化关系？它们使得我们可以利用继承的优点，促进了对象和程序代码的复用。

请注意图 9-22，当分析类图时，我们确定了 3 个泛化/特化层次：

建议的对象列表
ACTIVE MEMBER
BILLING ADDRESS
CLUB MEMBER
CREDIT CARD ACCOUNT
DISTRIBUTION CENTER
E-MAIL ADDRESS
MEMBER
MEMBER ORDER
PROMOTION
MEMBER ORDERED PRODUCT
PREORDER
PRODUCT
SHIPPING ADDRESS
加上
AGREEMENT
AUDIO TITLE
FORMER MEMBER
GAME TITLE
INACTIVE MEMBER
MERCHANDISE
RETURN
TITLE
TRANSACTION
VIDEO TITLE

图 9-20 会员服务系统建议的对象列表

	CLUB MEMBER	MEMBER ORDER	MEMBER ORDERED PRODUCT	PRODUCT
CLUB MEMBER		提交零个或多个	购买了零个或多个	XX
MEMBER ORDER	由一个且仅一个提交		包含零个或多个	XX
MEMBER ORDERED PRODUCT	由一个且仅一个购买	一个是 一个且仅一个的部分		与一个且仅一个相关
PRODUCT	XX	XX	销售零个或多个	

图 9-21 对象关联矩阵举例

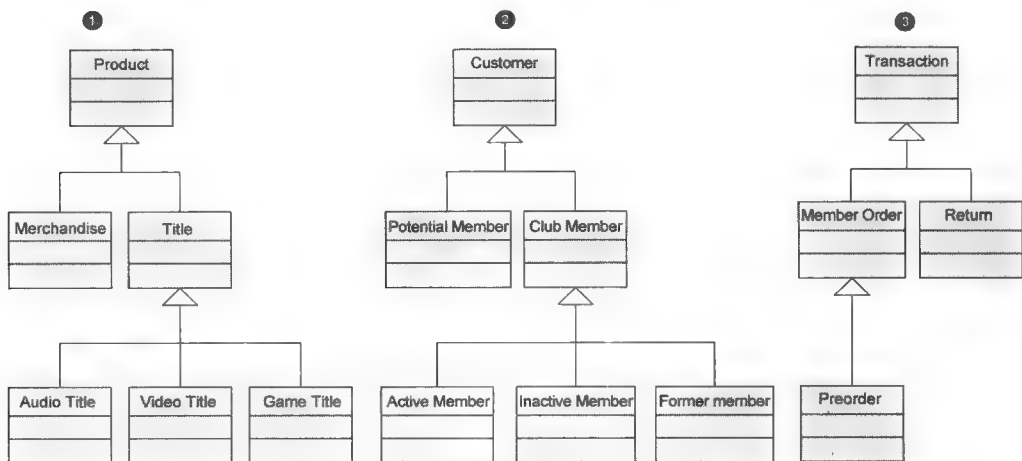


图 9-22 会员服务系统中的泛化/特化层次

❶“产品”层次使我们可以跟踪所有可购买的产品，并使我們能够在以后增加不同类型的产品，例如：图书。

❷“客户”层次使我们可以跟踪所有的客户（过去的、现在的和潜在的）。这允许我们发送特别促销给非活动的会员，鼓励他们再次开始订购产品。这使我们可以确定终止了会员关系的前会员，或者因为账号状态不良而被终止会员关系的前会员。并使我們能够在以后增加不同类型的客户，例如：公司客户。

❸“事务”层次使我们可以跟踪客户进行的各种事务。目前，记录了会员订单、预定和退款，但这个层次也可以修改以增加在以后发行的图书的预订。

9.4.8.3 第3步：确定聚合关系

在这一步，必须决定是否存聚合或组合关系。聚合是一类关联关系，其中一个对象是另一个对象的一部分。它经常被称为**整体/部分关系**，并且可以读做：对象 A 包含对象 B，并且对象 B 是对象 A 的部分。聚合关系是不对称的：对象 B 是对象 A 的部分，但对象 A 不是对象 B 的部分；聚合关系并不隐含继承：对象 B 没有从对象 A 继承属性和行为；但是，应用于整体的行为自动地应用于部分。例如，如果想将对象 A 发送到一个客户，则对象 B 也将被发送。

当分析类图时，我们确定了一个组合关系：“会员订单”和“订购的产品”之间的关系。

9.4.8.4 第4步：准备类图

图 9-23 是会员服务系统的一个部分 UML 类图^①。注意模型描述了音阶公司会员服务系统领域内的业务对象/类，模型中的对象/类符号没有描绘出行为（方法），这些内容将在第 17 章中确定和定义。

① 这个图使用 Popkin Software 公司的 System Architect 构造。

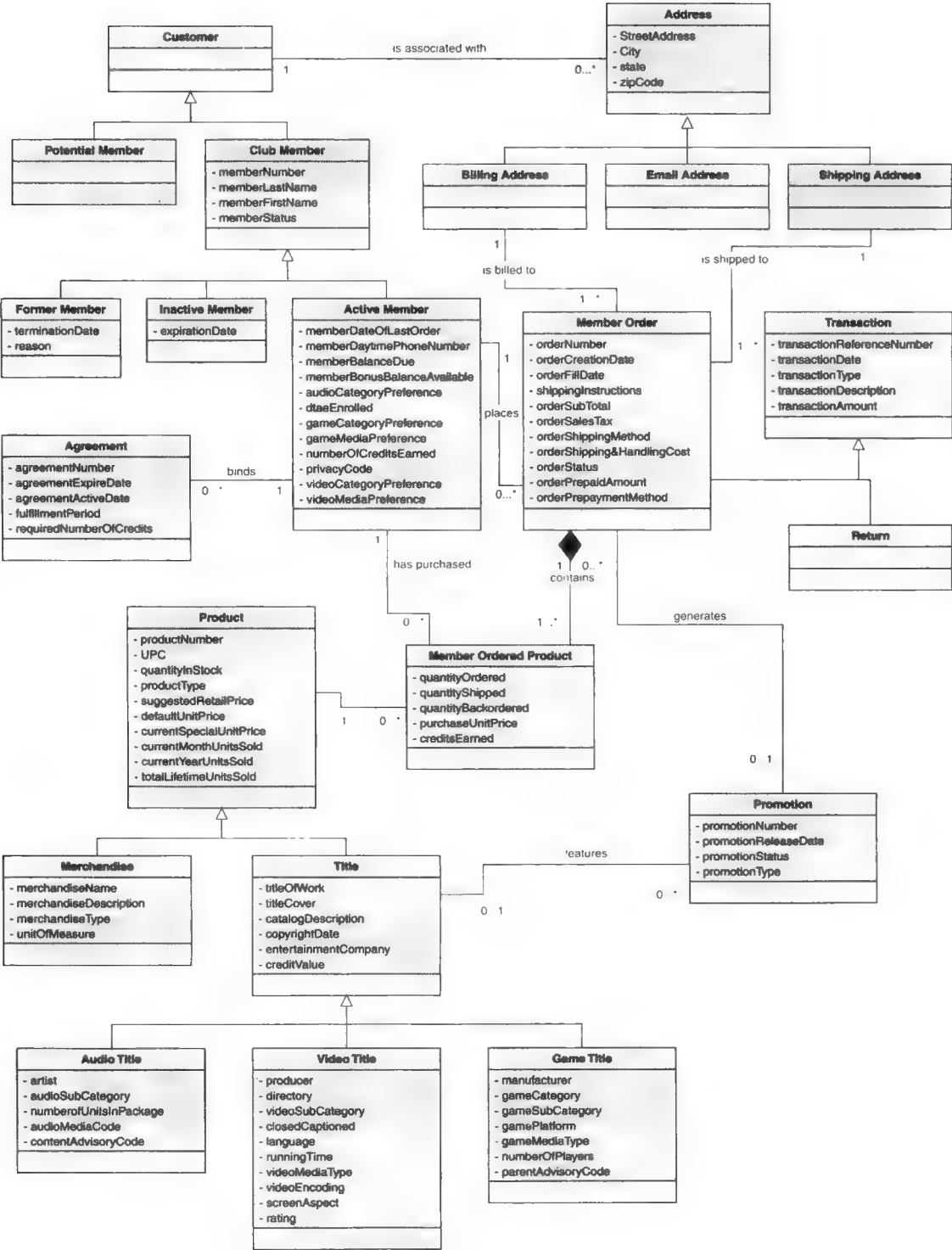


图 9-23 会员服务系统类图

这个模型图也反映了第1步中确定的类关联关系和多重性、第2步中发现的3个泛化/特化关系,以及第3步中发现的一个聚合(组合)关系,注意每个类符号的底部有一个词 *persistent* (持续的)。注意在类的底部的单词持续的,通常,这意味着类描述的对象将被永久存储在一个数据库中。所有的业务域类一般是持续的。由软件程序临时创建的对象称为临时对象。例如:对象实例的集合(如注册一门课程的所有学生),以便产生一份按字母顺序的学生花名册。一旦生成了花名册,程序就从内存中删除临时对象(集合)。临时对象通常在面向对象设计期间建模(见第17章)。

复习题

1. 对象建模最常被接受的记号标准是什么?
2. 对象被定义为“某种存在的,或者能被看到、触摸或以其他方式感觉到的事物,用户就该事物存储数据和相关行为”。请解释在这个定义中某种存在、数据和行为的含义。
3. 什么是封装?
4. 考虑课本和菜谱都是属于“书”类的对象。请给类及其对象列举一个例子。
5. 继承和超类/子类之间是什么关系?
6. 在面向对象分析和建模中,对象和类不是孤立存在的。为什么?
7. 分析员应该如何使用 UML 表示对象或类的关系?
8. 在表示聚合关系时,使用空心的菱形和使用实心的菱形之间有什么区别?
9. 什么是多态性,什么时候使用它?
10. 五组 UML 图形是什么?
11. 顺序图和协作图之间有什么区别?
12. 在进行面向对象分析中的三个主要活动是什么?
13. 什么是活动图?什么时候使用它?
14. 决定候选对象是否有用,是否应该被保留,或者是否应该被丢弃的方法有哪些?
15. 组织对象并确定它们之间的关系的步骤是什么?

问题和练习

1. 自从1997年被引入,统一建模语言(UML)在全世界快速地获得了广泛的接受和使用。
 - a. 在对象建模方面,UML给设计人员提供了什么?UML没有提供什么?
 - b. 开发UML的原因是什么?
 - c. 如果没有开发UML,今天的对象建模可能会是什么样子?
2. 面向对象分析(OOA)和对象建模在许多企业成了熟悉的词汇,但它们底层的含义并不总是那么明显,可能会难以理解,特别是对系统开发项目中涉及到的非技术用户来说。
 - a. 用非技术的词汇,解释什么是对象,什么是面向对象分析方法。
 - b. 另外用非技术的词汇,解释对象建模技术。
 - c. 面向对象分析和传统的系统分析在它们进行系统开发方面主要的不同点是什么?
 - d. 如果你是一位在传统开发方法方面有经验的系统设计人员,你认为学习面向对象分析方法容易吗?如果这是你学习的第一个分析方法呢?解释你的回答。
3. 考虑电影DVD作为对象的例子。
 - a. 使用课本中的词汇,电影DVD是哪类对象?
 - b. 电影DVD有哪些属性?
 - c. 电影DVD的对象实例是什么?
 - d. 在对象模型中用UML记号表示“电影DVD”类,如图9-2所示。包括类名、属性和行为。
- e. “电影DVD”类被认为是一个超类或子类吗?举例说明。
4. 对于本练习,考虑对象与狗的不同例子。
 - a. 狗是哪类对象?
 - b. 狗有哪些属性?
 - c. 使用图9-1的例子表示“狗”的一个对象实例。
 - d. “狗”类具有什么行为?
 - e. 在UML中表示“狗”类,以图9-2b为例。
5. 再次考虑“狗”类。
 - a. 提供5或6个“狗”类和“人”类之间的关联关系的例子。
 - b. 为“狗”类显示对象类关联关系和多重记号。
 - c. “狗”类可能存在哪类聚集关系。
6. 为了交互,对象和类可以相互发送消息。
 - a. 列举一个从对象类“狗”到“人”发送消息请求的例子,以及“人”的反馈行为。
 - b. 在消息发送中,发送对象不需要知道接收对象什么?
 - c. 为了能够在两个对象之间发送消息,需要存在什么?
7. 多态性是一个对理解面向对象分析很重要的概念。你需要向项目团队中的系统用户解释这个概念。用非技术性的词汇:

- a. 定义多态性的概念。
- b. 解释多态性如何同消息相关。
- c. 解释重载行为是什么。
8. 你正在讲授一门面向对象分析和设计方面的介绍性课程。请解释：
 - a. 不同组的 UML 图，每组图描述和/或建模了什么。
 - b. 用例建模确定了什么。
 - c. 在进行面向对象分析中三项主要任务是什么。
 - d. 业务需求用例模型如何以及为什么要被细化并改变成分析用例模型。
9. 在设计阶段，需要开发抽象用例描述和扩展用例描述。
 - a. 为什么用于抽象用例和扩展用例的描述有所不同？
 - b. 记录抽象用例和扩展用例的描述与记录常规的用例有什么不同？
 - c. 抽象用例会被单个用例调用吗？
 - d. 扩展用例可复用吗？
 - e. 扩展用例会被单个用例调用吗？
 - f. 抽象用例会被常规用例调用吗？
10. UML 活动图用于建模系统过程活动，帮助系统分析员可视化用例的流程。
 - a. 它们同流程图相比有什么区别？这种区别有什么用处？
 - b. 它们有什么相似处？
 - c. 活动图中的实心黑条代表什么？
11. 在面向对象分析和建模中，确定所有的潜在对象十分重要。正如一些专家所建议的那样，这可以通过检查需求文档查找所有名词的方法来实现，因为每一个名词都代表一个可能的对象。
 - a. 这个方法有什么问题？
 - b. 用例建模如何帮助确定潜在对象？
 - c. 一旦确定了潜在（候选）对象，每个潜在对象都应该成为一个实际对象吗？
 - d. 应该如何选择候选对象？
 - e. 如果候选对象实际上是一个属性怎么办呢？
12. 进行面向对象分析的最后一步是组织对象并确定它们之间的关系。
 - a. 类图显示了系统的动态结构还是静态结构？
 - b. 为什么对象之间的关联关系在定义多重性之前确定？
 - c. 对象类矩阵的用途是什么？
 - d. 如果有 72 个对象和类，矩阵中将有多少个空位？
13. 确定了对象之间的关联关系和多重性之后，你必须完成另外几个步骤，对象的组织才会被认为完成了？
 - a. 这些步骤是什么？
 - b. 为什么在设计阶段确定泛化/特化关系很重要？
 - c. 确定可能的泛化/特化关系有哪两种技术？
 - d. 泛化/特化关系与聚类关系之间的基本差别是什么？
 - e. 业务领域类可以包含一个瞬态对象吗？

项目和研究

1. 自从 1997 年被引入，统一建模语言（UML）很快成为了对象建模的公共接受的标准和广泛使用的工具。访问 www.omg.org 网站，这是对象管理组织（OMG）和 UML 标准组织的网站，浏览它的 UML 资源页面，以及到其他站点的链接，例如 IBM 和 Popkin 软件。
 - a. UML 最近的版本是什么？
 - b. 什么是对象管理组织？
 - c. 在阅读网站上的历史文章时，你认为 UML 这么快地成为对象建模的主导工具的原因是什么？
 - d. 阅读 UML 2.0 标准（可以免费下载或阅读）——关于这个新版本你发现的最有价值或者最有趣的是什么？
 - e. 在信息技术中使用的许多语言出现之后就消失了，但是某些语言，例如 COBOL，在它产生后几十年仍然被广泛使用。根据网站上可得到的文章，你认为 UML 的生命期会怎么样？为什么？
 - f. 目前 UML 存在新的或者正在出现竞争者吗？
2. 找到一个熟悉 UML 建模的系统设计人员，并同他交谈。
 - a. 这位设计人员使用 UML 做过哪类系统？
 - b. 这位设计人员通过 CASE 工具使用 UML 吗？如果是，使用的是哪种工具？
 - c. 这位设计人员最喜欢 UML 什么？
 - d. 这位设计人员最不喜欢 UML 什么？
 - e. 如果这位设计人员能够选择建模语言，他会选择 UML 吗？为什么？
 - f. 这位设计人员希望看到哪些特征增加到 UML 中？
3. 假设你正作为一名自由职业的系统设计人员工

作，并为一个案件跟踪系统做一些设计工作。这个案件跟踪系统是为一个专门从事民事案件的本地法律公司开发的。业务目标是实现一个系统跟踪民事案件，从这个法律公司开始诉讼直到案件最终判决。

你将期望在一个专门从事民事文件档案的法律公司中找到哪些主要对象和类？

- a. 描述每个类，包括它们的名字和属性，使用图 9-2 的例子。
 - b. 描述任何泛化/特化关系，使用图 9-4 的例子。
 - c. 确定对象/类关联关系，创建一个以图 9-5 为例的对象/类关联关系表。确保包括了每个关联关系的多重性。
 - d. 确定聚集关系，使用图 9-6 的例子准备一个聚集关系表。
4. 这个法律公司认可你的工作，并且想延长你的合同，以便你能继续设计这个案件跟踪系统。假定你协商了一个可接受的报酬，对于你下一步的工作：
- a. 使用图 9-11 的例子创建至少两个详细的用例描述。
 - b. 使用图 9-12 的例子创建一个抽象用例描述。

c. 使用图 9-13 的例子为每一个用例创建一个活动图。

5. 这时，你想确认你已经包括了每项内容，没有遗漏什么。所以进行你的下一步工作：

- a. 使用本书中描述的技术寻找潜在对象。
 - b. 使用图 9-16 的例子创建潜在（候选）对象清单，并决定每个候选对象是应该保留还是丢弃。
 - c. 使用图 9-17 的例子创建一个建议的对象清单。
 - d. 使用图 9-18 的例子创建一个对象关联关系矩阵，确定存在于对象和类之间的关联关系和多重性。
 - e. 你发现了你以前没有确定的对象和类吗？
6. 你就要完成这个法律公司的案件跟踪系统的面向对象分析和建模了。基于你在上一题的设计工作，你最后的任务是：
- a. 使用图 9-19 的例子创建一个泛化/特化层次图。
 - b. 创建一个类图，使用图 9-19 的例子。
 - c. 此时，如果你能在使用 UML 的面向对象分析和建模或者一种传统的结构化设计方法之间进行选择，你会选择哪一个？为什么？

小型案例

1. 利用目前你收集的信息和你对需求的评估，提出一个系统以满足部门的当前需要，以及未来的需求和机会。准备一篇论文，其中包括：条件背景、你提出的系统的概述、该系统特定的技术要求。论文不要超过 12 页（1.5 倍行距）。
2. 准备一份全面的可行性分析，包括对你在问题 1 中提出的系统的经济、运行、进度、法律和技术分析。你的分析不要超过 30 页（1.5 倍行距）。

团队和个人练习

1. 圆桌讨论：能够创建或实现某项技术和应该创建或实现某项技术是有巨大区别的。我们如何决定是否应该创建或实现某项技术（或信息系统）？
2. 团队练习：正如本问题所描述的，政府正在考虑开发并实现一个跟踪系统来跟踪受到政府助学金而上大学者的大学职业情况。关于这些有哪些道

3. 为问题 1 和问题 2 中的系统创建用例描述和用例图。确保你创建的用例是完整而且清楚的。记住，在现实生活中，系统分析员/设计人员经常不是开发系统的人，而实际上，团队很少一起全体开会。用例的清晰和完整是必需的。
4. 准备一个关于问题 1-3 的材料的汇报，并在课堂上进行汇报。利用有趣的汇报形式（例如视频、声音等等）。

德的、经济的和技术的问题？你认为政府应该做这件事吗？

3. 团队练习：通过视频挖掘相关有哪些法律的、道德的和技术的问题？你认为零售商店，例如 Gap（如例子中所讨论的），应该实现这样一个系统吗？

可行性分析和系统方案建议

本章概述和学习目标

优秀的系统分析员在建议任何改变之前会全面地评价替代方案。在本章中，你将学到如何在可行性评价准则（运行可行性、技术可行性、进度可行性和经济可行性）的基础上分析和记录那些替代方案。你还将了解到如何以书面报告和正式演示汇报的形式提出一个系统方案建议。本章将介绍以下内容：

- 确定系统生命周期中的可行性检查点。
- 确定各种系统方案。
- 定义和描述四种类型的可行性及其相应的评价准则。
- 使用经过时间调整的成本和收益进行各种成本效益分析。
- 为不同的读者编写合适的系统方案建议报告。
- 计划为系统所有者和用户做一个正式的演示报告。

本章关键术语

运行可行性（operational feasibility）是对方案满足确定的系统需求以解决问题和利用可见的机会的能力的度量。

文化（或者政治）可行性（cultural（or political）feasibility）是对方案在给定的企业文化下被接受程度的度量。

技术可行性（technical feasibility）是对一种特定技术方案的现实性以及技术资源和专家的可用性的度量。

进度可行性（schedule feasibility）是对项目时间表的合理性的度量。

经济可行性（economic feasibility）是对一个项目或方案的成本效益的度量。

法律可行性（legal feasibility）是对方案能否在现有的法律和合同义务内实现的度量。

固定成本（fixed cost）是有规则的但相对固定的费用。

变动成本（variable cost）是与某些使用因素成比例的费用。

有形收益（tangible benefit）是那些容易被量化的收益。

无形收益（intangible benefit）是指被认为难以量化或者不可能量化的收益。

投资回收分析（payback analysis）是一种用于确定投资是否可以收回以及何时收回的技术。

投资回收期（payback period）是产生的收益超过产生的成本之前所经历的时间。

现值（present value）是在未来任何时候1美元的当前价值。

投资回报率（ROI）分析（return-on-investment analysis）是一种比较替代方案或项目的终生收益率的技术。

净现值（net present value）是一种比较不同方案的年度贴现成本和收益的分析技术。

候选系统矩阵（candidate systems matrix）是用来记录候选系统之间异同点的工具。

可行性分析矩阵（feasibility analysis matrix）是用来评定候选系统的工具。

系统方案建议（system proposal）是被推荐系统的一份书面报告或演示汇报。

正式汇报（formal presentation）是用来兜售新想法并获得新系统认可的专门会议。

10.1 可行性分析和系统方案建议

在如今的商业界中，分析员必须学会像企业经理一样进行思考。计算机应用在以一种前所未有的速度发展，现在比以前更是有过之而无不及，所以管理层希望信息系统能够收回投资。信息是一个必须经过检验的重要资本投入，就像市场要检验一个新产品，生产部门要检验一个新厂房或设备一样。

系统分析员比以前更多地被要求回答下面的问题：投资能够收回吗？是否有其他投资能够带来比预期更高的回报？

本章解决信息系统的系统分析员和用户感兴趣的可行性分析问题，并强调以系统方案建议的形式向管理层进行推荐的重要性，方案建议可以是一份正式的书面报告和/或口头汇报。可行性分析适用于系统分析阶段，但对决策分析阶段尤其重要。系统方案建议是决策分析阶段的交付成果，表示了技术性的“知识”、“过程”和“通信”方案。

10.1.1 可行性分析——逐步投入法

首先介绍可行性和可行性分析的正式定义。可行性是对组织将要开发的信息系统的价值或实用性的度量。可行性分析是度量可行性的过程。

在整个生命周期内都应该度量可行性，在前面的章节中，我们称之为逐步投入法。在初始问题和机会被全面地研究以后，或者当系统被设计之后，一个明显可行的项目的范围和复杂性也可能会发生变化。因此，一个以前可行的项目后来可能会变得不可行。

图 10-1 显示了系统生命周期中的系统分析阶段期间的可行性检查点，检查点用菱形表示。菱形指示一个可行性再评估和管理检查应该在前一阶段的结尾（下一阶段开始之前）进行。一个项目可以在任何检查点上被取消或者被修改，无论已经投入了多少资源。

这个想法最初可能会使你担心，你的自然倾向很可能是根据你已经花费的时间和投资证明应该继续一个项目。那些成本都是过去的了！一个基本的管理学原理就是从来不要为了补偿损失反而损失更多——应该减少你的损失，并转移到一个更可行的项目上去。这并不意味着已经投入的成本不重要，如果一个投资被认为是成功的话，最终成本必定会被收回。下面简要地介绍图 10-1 中的检查点。

10.1.2 系统分析——范围定义阶段的检查点

第一个可行性分析在范围定义阶段进行。在项目的这个早期阶段，可行性一般只是对问题紧急程度的度量以及对开发费用的粗略估计。它回答以下问题：问题（或机会）值得投入对当前系统的详细研究和分析的费用吗？实际上，除非问题（或机会）和需求被更好地理解，否则可行性不可能被正确地度量。

在估计了解决问题和机会的收益后，分析员将估计开发预期系统的成本。有经验的分析员一般将成本调高 50% ~ 100%（或者更多），因为经验告诉他们问题很少能够被充分定义，而且用户需求一般很少能够被全面地理解。

10.1.3 系统分析——问题分析阶段的检查点

下一个检查点出现在对当前系统更详细的研究和问题分析之后。因为问题已经被更好地理解，所以分析员可以更好地估计开发成本和从新系统中获得的收益。解决一个问题的最低评估等于那个问题的成本。例如，如果库存运输成本的可接受上限为 35 000 美元，那么一个可接受的信息系统的最低价值就是 35 000 美元。我们当然希望一个改进后的系统能够做得更好，但是，它必须至少返回这个最低价值。

在这一点上，开发成本仍只是一个估计值。我们仍需要全面地定义用户需求或者详细说明那些需求的设计方案。

如果从范围定义阶段到问题分析阶段估计的成本明显地增大，那么问题可能出在项目范围上。在许多项目中，范围总是倾向于增加。如果范围的增加威胁到可行性，那么就应该减小范围。

10.1.4 系统设计——决策分析阶段的检查点

决策分析阶段代表了一个主要的可行性分析活动，因为它从许多可能的实现中选择一个作为系统设计的目标。

现在问题和需求应该已经明确。在决策分析阶段期间，替代方案按照其输入/输出方法、数据存储方法、计算机硬件和软件需求、处理方法以及人员情况进行定义。下面的清单表示可供分析员评估的典型选择范围。

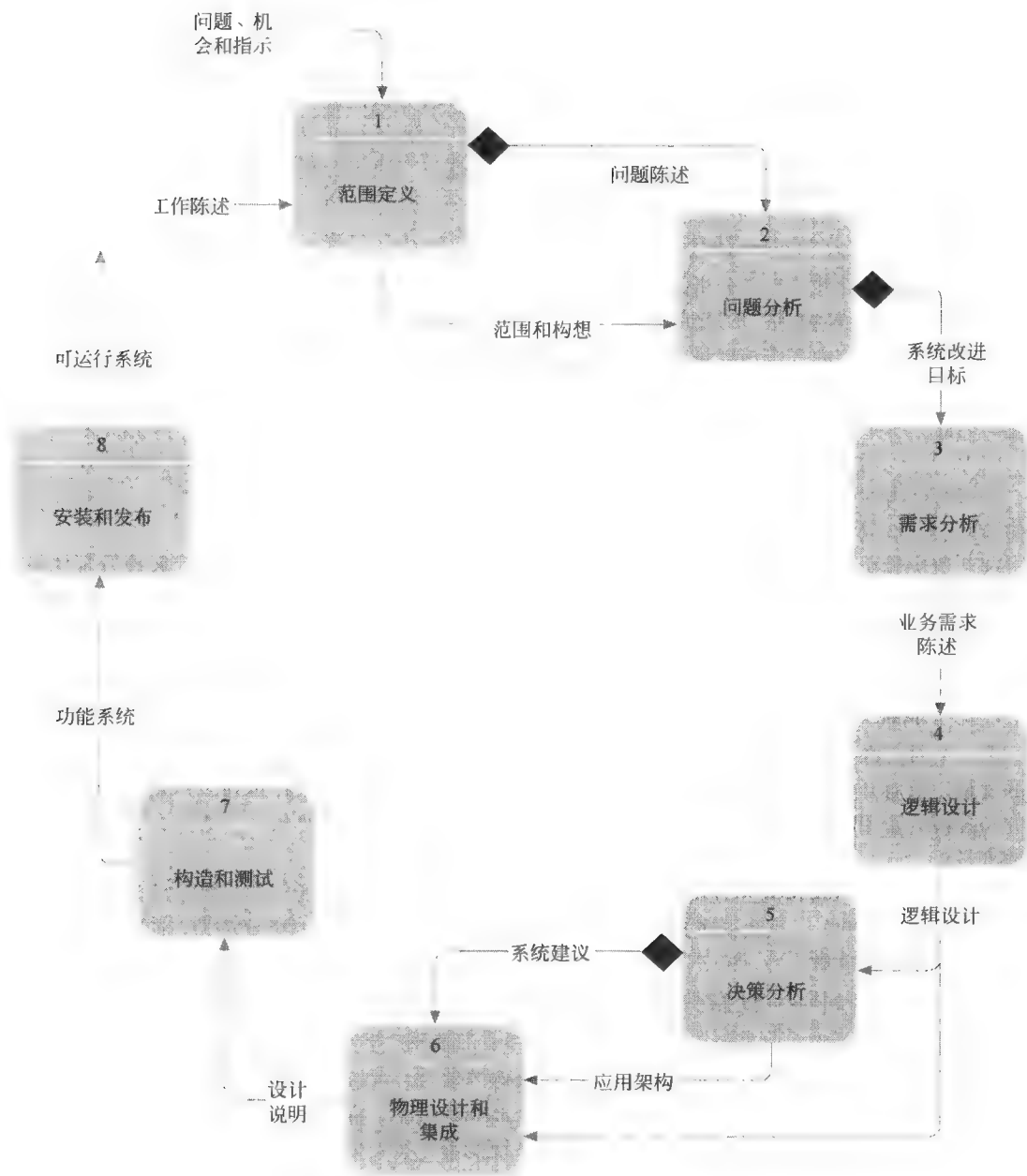


图 10-1 系统分析期间的可行性检查点

- 什么也不做！保持当前系统不变。无论管理人员或者你自己对这种选择的观点如何，它都应该作为一个基线被考虑和分析，所有其他的方案能够并且应该按照这个基线进行评价。
- 再工程（手工的）业务过程，注意不是基于计算机的过程。其中包括理顺活动、减少重复和不必要的任务、重新组织办公室布局、消除冗余和不必要的表格和规程，以及其他内容。
- 改进现有计算机过程。
- 购买一个封装式应用系统。
- 设计并构造一个新的计算机系统。

定义了这些选项后，分析每个选项的运行可行性、技术可行性、进度可行性和经济可行性。本章将深入地介绍这 4 类可行性评价准则。一种替代方案将被推荐给系统所有者作为方案建议，被建议的

方案将成为概要设计和详细设计的基础。

10.2 可行性的 6 个准则

到目前为止，我们已经定义了可行性和可行性分析，而且已经确定了系统分析期间的可行性检查点。可行性可以从多个角度来查看。下面介绍 6 种可行性准则。

- **运行可行性**度量方案满足确定的系统需求以解决问题和利用可见的机会的程度。
- **文化（或者政治）可行性**是对人们对方案的感觉以及方案在给定的企业文化下被接受程度的度量。
- **技术可行性**是对一种特定技术方案的现实性以及技术资源和专家的可用性的度量。
- **进度可行性**是对项目时间表的合理性的度量。
- **经济可行性**是对一个项目或方案的成本效益的度量。
- **法律可行性**是对方案能否在现有的法律和合同义务内实现的度量。

实际上，很少有系统是完全不可行的，相反，不同的方案往往在某些方面比较可行。下面我们更详细地介绍这些可行性。

10.2.1 运行可行性

运行可行性是对一个建议的方案解决在范围定义和问题分析阶段确定的问题和利用可见的机会的能力的度量，以及它对需求分析阶段确定的系统需求的满足度。运行可行性也会询问：给定现在对问题的理解和方案的费用，问题仍值得去解决吗。PIECES 框架（见第 2 章）可以用作分析一个问题的紧急程度或者一个方案的有效性的基础。

10.2.2 技术可行性

如今，很少有技术上不可能的事情。所以，技术可行性主要考虑技术是否实际和合理。技术可行性涉及以下三个主要问题：

1. 建议的技术或方案实际吗？
2. 我们目前拥有所需的技术吗？
3. 我们拥有所需的技术专家吗？进度合理吗？

10.2.2.1 建议的技术或方案实际吗

方案中使用的技术通常是可得到的，但问题是技术是否足够成熟，是否方便地应用到我们的问题上。有些公司喜欢使用最新技术，但大多数公司喜欢使用成熟的经过了考验的技术。一种成熟的技术有较大的客户基础，以获得有关问题和改进的建议。

10.2.2.2 我们目前拥有所需的技术吗

假定方案要求的技术是实际的，我们必须再次自问，我们的信息系统部门拥有该项技术吗？如果可得到相应技术，我们必须再问我们是否具有相应的能力。例如，我们当前的打印机能够处理一个新系统要求的新报告和表格吗？

如果这些问题中任何一个的答案为否，那么我们必须自问，我们可以得到这个技术吗？技术可能是实际的和可得到的，而且我们确实需要它，但我们可能不能及时地提供这个技术。尽管这个论点与经济可行性近似，但它确实是技术可行性问题。如果我们不能提供该技术，那么需要该技术的方案是不现实的并且技术上不可行。

10.2.2.3 我们拥有所需的技术专家吗

在可行性分析期间，有关技术可行性的考虑经常被忽略。我们可能拥有这项技术，但这并不意味着我们拥有正确地应用该项技术所需的技能。例如，我们可能有一个数据库管理系统（DBMS）。但是，该项目可用的分析员和程序员对那个 DBMS 的了解程度可能不足以正确地运用它。确实，所有的信息系统专家都可以学习新技术，但是学习曲线将影响项目的技术可行性，特别地，它将影响进度。

10.2.3 进度可行性

指定技术专家后，项目的最后期限合理吗？也就是说，项目的进度可行性如何？有些项目由特定

的最后期限引发，你需要确定最后期限是强制的还是期望的。例如，开发一个新系统以满足新政府报告规定的项目可能有一个最后期限，这个期限正好同新报告必须启用的时间吻合，错过这样一个最后期限导致的相关处罚可能使这个最后期限成为强制的。如果最后期限是期望的而不是强制的，分析员可以建议替代的进度方案。

与其按照期限要求发布一个易出错的、无用的信息系统，不如推迟两个月发布一个正确的、有用的信息系统（除非最后期限是绝对强制的）！延误进度是不好的，但不完善的系统是更大的错误。这是两劣取其轻。

10.2.4 经济可行性

许多项目的底线是经济可行性。在项目的早期阶段，经济可行性分析最多不过是判断解决问题的可能效益是否值得。因为最终用户需求和替代技术方案还没有得到确定，所以成本在这个阶段实际上是不可能估计出来的。但是，一旦确定了特定的需求和方案，分析员就可以权衡每个替代方案的成本和效益，这称为成本效益分析，将在本章后面讨论。

10.3 成本效益分析技术

经济可行性已经被定义为一种成本效益分析。如何估计成本和收益？如何比较那些成本和收益以确定经济可行性？大多数学校提供关于这些主题的完整课程——财务管理、财务决策分析以及工程经济学和分析的课程，这样一门课程应该包括在你的学习计划中。

10.3.1 系统将花费多少

成本分为两类：同开发系统相关的成本和同运行系统相关的成本。前者可以从一个项目的开始就进行估计，并且应该在项目的每个阶段的结尾进行精炼；后者只能在特定的计算机方案被定义后才能被估计出来。让我们更深入地看看信息系统的成本。

开发信息系统的成本可以按照成本出现的阶段分类。系统开发成本通常是一次性成本，在项目完成之后不会重现。许多组织具有必须被评估的标准成本分类项目，如果没有这种分类，可参考以下清单：

- 人工成本——系统分析员、程序员、咨询顾问、数据录入人员、计算机操作员、秘书等项目工作人员的薪水构成了人工成本。因为这些人中有许多人同时在多个项目中工作，所以他们的薪水应该按比例进行分配，以反映他们花在正被估计的项目上的时间。
- 计算机使用——计算机将被用于一个或者多个以下的活动中：编程、测试、转换、字处理、维护项目字典、原型化、加载新数据文件等等。如果计算中心对计算机资源的使用收费（例如对磁盘存储或报告打印收费），就应该考虑这项成本。
- 培训——如果计算机人员或最终用户需要培训，培训课程可能需要费用。封闭式培训课程可能按照每个培训点收取固定费用、按照每个学生收费（例如每个学生 395 美元）或者按照每小时收费（例如每课时 75 美元）。
- 供应、复制和设备成本。
- 任何新计算机设备和软件的成本。

图 10-2 的例子显示了一个典型方案的开发成本。当估计开发费用时，为系统运行后仍将发生的费用留出经费尤为重要。生命期的收益必须超过开发成本和运行成本。与系统开发成本不同，运行成本往往贯穿整个系统生命期。在它有效的生命期内，运行系统的成本可以按固定成本和可变成本进行分类。

固定成本是有规则的但相对固定的费用。固定运行成本的例子包括：

- 租借费用和软件许可证费用。
- 信息系统操作员和支持人员的按比例分配的工资（尽管薪水趋于增加，但增加是逐渐的而且往往不会从一个月到另一个月发生很大的变化）。

变动成本是与某些使用因素成比例的费用。例子包括：

- 计算机使用费用（例如，使用的 CPU 时间、使用的终端连接时间和使用的存储空间），它根据工作负载而变化。

客户/服务器系统方案的估计成本

开发成本

人员:

2	系统分析员 (每人 400 小时, 每小时 50.00 美元)	40 000 美元
4	程序员/分析员 (每人 250 小时, 每小时 35.00 美元)	35 000 美元
1	GUI 设计人员 (每人 200 小时, 每小时 40.00 美元)	8 000 美元
1	通信专家 (每人 50 小时, 每小时 50.00 美元)	2 500 美元
1	系统架构师 (每人 100 小时, 每小时 50.00 美元)	5 000 美元
1	数据库专家 (每人 15 小时, 每小时 45.00 美元)	675 美元
1	系统资料员 (每人 250 小时, 每小时 15.00 美元)	3 750 美元

花费:

4	Smalltalk 培训费 (每个学生 3 500.00 美元)	14 000 美元
---	----------------------------------	-----------

新硬件和软件:

1	开发服务器	18 700 美元
1	服务器软件 (操作系统等)	1 500 美元
1	DBMS 服务器软件	7 500 美元
7	DBMS 客户端软件 (每个客户 950.00 美元)	6 650 美元

总开发成本:

143 275 美元

预计的年运行成本

人员:

2	程序员/分析员 (每人 125 小时, 每小时 35.00 美元)	8 750 美元
1	系统资料员 (每人 20 小时, 每小时 15.00 美元)	300 美元

花费:

1	服务器维护合同	995 美元
1	服务器 DBMS 软件维护合同	525 美元
	预打印的表格 (每年 15 000 张, 每张 0.22 美元)	3300 美元

预计的总年度成本:

13 870 美元

图 10-2 一个建议的系统方案的成本

- 供应 (例如, 预打印的表格、使用的打印纸、穿孔卡片、软盘、磁带和其他花费), 它根据工作负载而变化。
- 按比例分配的管理费用 (例如, 公共设备、维护和电话服务), 它可以使用标准的成本统计技术在整个系统生命期分配。

图 10-2 的例子也显示了一个方案的运行成本估计。

10.3.2 系统将提供什么收益

收益一般会增加利润或者降低成本, 这些都是一个新信息系统所期望的特征。收益应该尽可能地以货币来量化, 它可以分为有形收益或无形收益。

有形收益是那些可以进行量化的收益。有形收益按照公司月度或者年度积余或者利润的形式度量。例如, 考虑以下场景:

处理学生住房应用系统需要重复输入和填写大量的数据。一次分析揭示出同一数据被输入了 7 次, 平均每个应用需要办事员多花费 44 分钟工作时间。办公室每年处理 1 500 个应用。这意味着每年总共 66 000 分钟或 1 100 小时的重复工作。如果一个秘书的平均薪水是每小时 15 美元, 则这个问题的成本和解决这个问题的收益就是每年 16 500 美元。

另外, 有形收益可以按照单位成本积余或利润的形式来度量。例如, 库存估价方案可以减少库存运输成本每库存单位 0.32 美元。有形收益包括: 较少的处理错误、增加的吞吐量、减少的响应时间、工作步骤的精简、增加的销售、信用损失降低、成本减少。

另一类收益是无形的。**无形收益**是那些被认为难以量化或者不可能量化的收益。如果没有这些收益,许多项目可能完全不可行。无形收益包括:改善的客户亲切感、提高的雇员士气、对社区服务的更好服务、更好的决策。

遗憾的是,如果一个收益不能被量化,就难以接受一个建立在不完整数据上的成本效益分析的有效性。有些分析员怀疑无形收益的存在性,他们认为所有的收益都是可量化的:一些收益只是比其他收益更难以量化。例如,假设改善的客户亲切感被列为一种可能的无形收益。我们可以量化亲切感吗?你可以试试以下的分析:

1. 客户不高兴的结果将是什么?客户将发出更少的(或者不发出)订单。

2. 客户订单将减少到什么程度?你的用户可能觉得难以明确地量化这种影响,但你可以试着让最终用户估计可能性(或者发明一个最终用户可以回答的估计),例如:

a. 固定客户有 50% 的可能将发送较少的订单——少于所有订单的 10%——给竞争者,以测试竞争者的性能。

b. 固定客户有 20% 的可能将发送一半的订单给竞争者,特别是那些以往履行得很慢的订单。

c. 固定客户有 10% 的可能将给我们发送最后一份订单作为最后的手段,这将减少客户同我们的常规业务到当前量的 10% (90% 或 0.90 的损失)。

d. 固定客户有 5% 的可能将选择根本不同我们做业务 (100% 或 1.00 的损失)。

3. 我们可以计算出估计的业务损失,如下:

损失 = $0.50 \times (0.10 \text{ 业务损失}) + 0.20 \times (0.50 \text{ 业务损失}) + 0.10 \times (0.90 \text{ 业务损失}) + 0.50 \times (1.00 \text{ 业务损失}) = 0.29 = 29\%$ 的业务损失统计估计

4. 如果客户平均每年业务量为 40 000 美元,那么预计损失业务量的 29% (或者 11 600 美元)。如果有 500 个客户,则预期业务损失总额将为 5 800 000 美元。

5. 向管理人员报告这个分析结果,并以它作为量化收益的初始值。

10.3.3 建议的系统合算吗

有一种常用的技术可用于评估经济可行性,也称为成本效益:投资回收分析、投资回报率和净现值。

技术的选择应该考虑使用技术的人,几乎所有通过商业学校学习的经理都熟悉这三种技术。每种技术都要使用的一个概念是调整成本和收益以反映货币时间价值。

10.3.3.1 货币时间价值

所有三种技术共同使用的一个概念是**货币时间价值**——今天的 1 美元比一年后的 1 美元更值钱。可以今天投资 1 美元,然后通过应计利息,从现在起一年后会超过 1 美元。因此,今天拥有这 1 美元比在一年后拥有更好。这就是为什么债权人想让你尽快地支付账单——他们不能投资他们没有的东西。在进行成本效益分析之前,同样的原理可以应用于成本和收益。

系统的某些成本在实现后会自然地生成,而新系统的所有收益也将在未来自然地生成。在进行成本效益分析之前,这些成本应该被换算成当前的美元值。举例有助于说明这个概念。

假设将从现在开始两年后实现 20 000 美元的收益。两年后,20 000 美元收益的当前美元价值是多少?收益的当前价值是我们需要今天投资从现在起两年后得到 20 000 美元的货币量。如果当前投资回报率是 10%,今天投资 16 528 美元将在两年后变成 20 000 美元(后面我们将演示如何计算)。所以,估计收益的当前价值是 16 528 美元——也就是说,宁愿今天拥有 16 528 美元,而非从现在起两年后承诺的 20 000 美元。

因为项目经常要与具有不同生命期的其他项目进行比较,所以时间价值分析技术已经成为大多数经理首选的成本效益分析方法。通过使用时间调整的成本和收益,你可以进行以下的成本效益分析技术。

10.3.3.2 投资回收分析

投资回收分析技术是一种简单而流行的方法,用于确定投资是否可以收回以及何时收回。因为系统开发成本在收益开始出现之前就已经产生很长时间,所以收益超过成本需要一些时间。在系统实现之后,会产生必须克服的额外运行成本。投资回收分析决定产生的收益需要多长时间才能超过已产生

的和继续产生的成本。这个时间段称为**投资回收期**。

在图 10-3 中, 我们看到将要花费 418 040 美元进行开发一个信息系统。6 年内每年净运行成本的估计值也在表格中记录, 在同样的 6 个运行年中估计的净收益也在表中显示。投资回收期是多长?

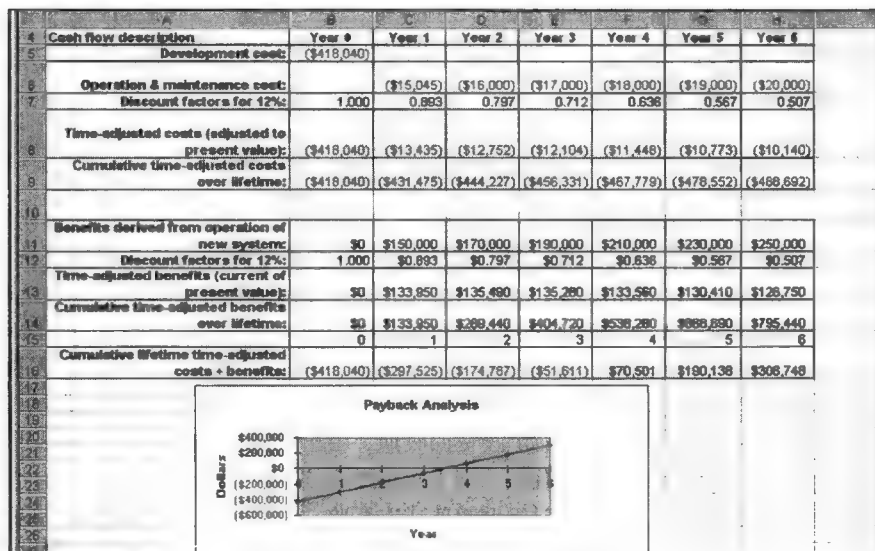


图 10-3 一个项目的投资回收分析

首先, 我们需要调整成本和收益的货币时间价值 (即把它们调整成为当前的美元值)。下面就是计算过程: 1 美元在第 n 年的现值取决于贴现率。贴现率有点类似于你从你的存款账号中获得的利息率。在大多数情况下, 一个企业的贴现率是能够投资到其他项目的机会成本, 这包括投资到股票、基金、债券等的可能性。另外, 贴现率可以表示公司认为可接受的投资回报率有多少。这个数字可以通过询问财务经理、官员或审计员了解到。

假设例中公司的贴现率是 12%。在未来任何时候 1 美元的当前价值——实际上称为**现值**——可以使用下面的公式计算:

$$PV_n = 1/(1+i)^n$$

其中, PV_n 是从现在起第 n 年 1.00 美元的现值, i 是贴现率。所以, 从现在起两年后 1 美元的现值是:

$$PV_2 = 1/(1+0.12)^2 = 0.797$$

前面我们说过今天的 1 美元比从现在起一年后的 1 美元值钱, 但是看起来好像它的价值减少了。这是一种错觉, 现值要按照如下方式解释: 如果你今天有 79.7 美分, 要比你两年后拥有 79.7 美分要多, 多多少呢? 确切地说要多 20.3 美分, 因为 79.7 美分将在两年内增长到 1 美元 (假定 12% 的贴现率)。

为了确定第 2 年成本或收益的现值, 只需简单地把估计的成本或收益乘上 0.797 即可。例如, 估计第 2 年运行成本是 16 000 美元, 这个成本的现值就是 16 000 美元 \times 0.797, 或者 12 752 美元 (四舍五入)。幸运的是, 你不必计算贴现因子。有专门的表格显示了不同时间段和贴现率下 1 美元的现值, 如图 10-4 所示。只需简单地将估计的成本或收益乘上这个数就可以得到其现值。这张表更详细的版本可以在许多会计和财务课本以及电子表格软件中找到。

大多数电子表格软件包括了内建的函数用于计算任何现金流 (成本或收益) 的现值。本节的所有例子都是使用 Excel 制作的, 这些表格也可以用 Lotus 1-2-3 制作。电子表格软件的好处在于一旦确定了行、列和函数, 只需要简单地输入成本和收益, 电子表格软件就将自动把这个数贴现为其现值 (实际上, 你也可以对电子表格编程, 以实施成本效益分析)。

回到图 10-3, 我们已经把例中的所有成本和收益换算为现值。注意第 0 年的贴现率是 1.000, 为什么呢? 1 美元在第 0 年的现值就是 1 美元, 这是有意义的, 如果你今天拥有 1 美元, 它就值 1 美元。

年	8 %	9 %	10 %	11 %	12 %	13 %	14 %
1	0.926	0.917	0.909	0.901	0.893	0.885	0.877
2	0.857	0.842	0.826	0.812	0.797	0.783	0.769
3	0.794	0.772	0.751	0.731	0.712	0.693	0.675
4	0.735	0.708	0.683	0.659	0.636	0.613	0.592
5	0.681	0.650	0.621	0.593	0.567	0.543	0.519
6	0.630	0.596	0.564	0.535	0.507	0.480	0.456
7	0.583	0.547	0.513	0.482	0.452	0.425	0.400
8	0.540	0.502	0.467	0.434	0.404	0.376	0.351

图 10-4 1 美元的现值

现在我们已经贴现了成本和收益，可以完成投资回收分析，这需要考虑累计的终生成本和收益。终生成本 6 年期间逐渐地增加，因为运行成本在累积。但也要注意终生收益在以更快的速度增加，终生收益将在第 3 年和第 4 年之间超过终生成本。通过表格化经过时间调整的累积终生成本 + 收益，我们可以估计出收支平衡点（当成本 + 收益 = 0 时）将在系统开始运转后大约 3.5 年的时候出现。

这个信息系统是个好的投资还是个差的投资？很难说！许多公司拥有一个对所有投资项目的投资回收期指南。在没有这类指南的情况下，你需要在确定投资回收期之前确定一个合理的指南。假设指南规定所有的投资必须具有少于或等于 4 年的投资回收期，由于我们的例子中投资回收期是 3.5 年，所以它是一个好的投资。如果系统的投资回收期大于 4 年，那么信息系统将是一个不利的投资。

应该注意到，你也可以使用没有经过时间调整的成本和收益进行投资回收分析，但结果将是 2.8 年的投资回收期，这看上去比我们计算的 3.5 年的投资回收期更诱人。因此，没有经过时间调整的投资回收期往往过于优化而具有误导性。

10.3.3.3 投资回报率分析

投资回报率 (ROI) 分析技术比较替代方案或项目的终生收益率。一个方案或者项目的 ROI 是度量企业从一项投资中获得的回报总量与投资总量之间关系的百分率。一个潜在的方案或项目的终生 ROI 计算如下：

$$\text{终生 ROI} = (\text{估计的终生收益} - \text{估计的终生成本}) / \text{估计的终生成本}$$

让我们对用来讨论投资回收分析时的同一个方案计算终生 ROI。同样，所有的成本和收益应该是经过时间调整的，经过时间调整后的成本和收益在图 10-3 的第 9 行和第 16 行显示。估计的终生收益减去估计的终生成本等于：

$$795\,440 \text{ 美元} - 488\,692 \text{ 美元} = 306\,748 \text{ 美元}$$

所以，终生 ROI 等于：

$$\text{终生 ROI} = 306\,748 \text{ 美元} / 488\,692 \text{ 美元} = 0.628 \approx 63\%$$

这是一个终生 ROI，而不是年度 ROI，简单地除以系统的生命期 (63/6) 就得到平均 ROI 为每年 10.5%。这个方案可以与其他方案进行比较，提供了最高 ROI 的方案是最佳方案。但是，与投资回收分析的情况一样，企业可能会为所有的投资设置一个最小可接受的 ROI。如果没有一个方案满足或者超过那个最小标准，那么就没有一个方案是经济上可行的。电子表格软件可以通过其内建的财务分析功能极大地简化 ROI 分析。

我们可以不用经过时间调整的成本和收益计算 ROI。但是，这将导致一个具有误导性的结果，即终生 ROI 为 129.4% 或者年度 ROI 为 21.6%。因此，建议把所有的成本和收益时间调整到当前的美元价值。

10.3.3.4 净现值

许多管理人员将投资的净现值作为首选的成本效益分析技术，特别是那些受过扎实的商业学校教育的人。首先需要确定系统的生命期间每年的成本和收益，而且需要调整所有的成本和收益，换算成现在美元价值。

图 10-5 说明了净现值技术，其中费用被表示为负的现金流，而收益被表示为正的现金流。我们已经把例中的所有成本和收益换算成现值。再次注意第 0 年的贴现率（用来累计所有的开发成本）是 1.000，因为 1 美元在第 0 年的现值就是 1 美元。

	A	B	C	D	E	F	G	H	I	J
1	Net Present Value Analysis for Client-Server System Alternative									
2	(Numbers rounded to nearest \$1)									
3										
4	Cash flow description	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Total	
5	Development cost:	(\$418,040)								
6	Operation & maintenance cost:		(\$15,045)	(\$16,000)	(\$17,000)	(\$18,000)	(\$19,000)	(\$20,000)		
7	Discount factors for 12%:	1.000	0.893	0.797	0.712	0.636	0.567	0.507		
8	Present value of annual costs:	(\$418,040)	(\$13,435)	(\$12,752)	(\$12,104)	(\$11,448)	(\$10,773)	(\$10,140)		
9	Total present value of lifetime costs:									(\$488,692)
10										
11	Benefits derived from operation of new	\$0	\$150,000	\$170,000	\$190,000	\$210,000	\$230,000	\$250,000		
12	Discount factors for 12%:	1.000	\$0.893	\$0.797	\$0.712	\$0.636	\$0.567	\$0.507		
13	Present value of annual benefits:	\$0	\$133,950	\$135,490	\$135,280	\$133,560	\$130,410	\$126,750		
14	Total present value of lifetime benefits:									\$795,440
15										
16	NET PRESENT VALUE OF THIS ALTERNATIVE:									\$306,748
17										

图 10-5 一个项目的净现值分析

在贴现了所有的成本和收益之后，将贴现后的收益减去贴现后的成本就等于净现值。如果结果为正，投资就是好的；如果为负，投资就不够好。当比较多个方案或项目时，具有最高正净现值的方案是最佳投资方案（即使替代方案具有不同的生命期，这个技术也可行）。在我们的例子中，被评估的方案得到一个净现值 306 748 美元，这意味着如果在 12% 的贴现率下我们投资 306 748 美元，6 年后我们将得到与实现这个信息系统方案获得的相同的利润。如果没有其他的方案具有超过 306 748 美元的净现值的话，这个方案就是一个好的投资。

电子表格软件通过其内建的财务分析功能可以极大地简化净现值分析。

10.4 候选系统的可行性分析

在系统分析的决策分析阶段，系统分析员确定了候选系统方案，然后分析这些方案的可行性。在本章中我们讨论了用于分析的评价准则和技术。本节将评价两个可以极大地促进候选系统方案的比较和对比的归档技术。二者都使用了矩阵格式。我们发现这些矩阵对向管理人员介绍候选方案和建议十分有用。

10.4.1 候选系统矩阵

第一个矩阵使得我们可以比较候选系统的几个特征。候选系统矩阵记录了候选系统之间的异同点，但是，它没有提供分析。

矩阵的列表示候选方案。优秀的分析员总是考虑多个实现选择，那些选项中至少有一个应该是现有系统，因为它将作为我们比较替代方案的标准。

矩阵的行表示候选方案的特征。根据本书的目的，我们把这些特征建立在信息系统构件基础之上，具体分解如下：

- 关联人员——确定系统将如何与人以及其他系统交互。
- 知识——确定数据存储将如何实现（例如，传统文件、关系数据库和其他数据库结构）；如何收集输入（例如，联机、批处理等）；如何产生输出（例如，按进度、根据请求、打印、在屏幕上显示等）。
- 过程——确定如何修改手工的业务过程，如何实现计算机过程。对于后者，我们有多项选择，包括联机处理与批处理、封装式软件包与内部构造软件。
- 通信——确定如何分布过程和数据。我们可以考虑几种替代方案——例如，集中方案、分散方案、分布（或重复）方案、协作（客户/服务器）方案。网络分布类型和策略将在第 12 章讨论。矩阵的单元格记录了哪些特征有助于读者理解选项之间的差别。图 10-6 演示了矩阵的基本结构。

	候选方案 1	候选方案 2	候选方案 3
关联人员			
知识			
过程			
通信			

图 10-6 候选系统矩阵模板

在考虑任何方案之前，我们必须考虑方案的约束条件。方案的约束条件采取架构决策的形式表现，目的是给应用系统带来秩序和一致性。例如，技术架构可能将方案限制在关系型数据库或客户/服务器网络。

有几种方法可用于确定候选方案，其中包括：

- 认识用户表达的想法和观点。在一个系统项目中，用户可以建议手工的或者与技术相关的方案。无论这些建议显得多么低效，都应该考虑。
- 咨询方法学和架构标准。许多组织的开发方法学和架构标准可以指示如何选择技术方案以及可以使用什么技术。
- 集体讨论可能的方案。集体讨论是一种确定可能方案的有效技术。使用一个有组织的方式或者框架进行集体讨论特别有效，例如采用 IS 构件或者其他 IS 特征。集体讨论应该包含那些代表了购买、构造以及组合购买与构造的方案。
- 寻找参考资料。分析员应该从其他实现过类似系统的人和组织那里征求意见和观点。
- 借阅相应的杂志和期刊。这类文献可能会包含关于自动化策略、成功、失败和技术的广告和文章。

开发团队成员可独立地使用以上方法的组合来得到一些可能的替代系统方案。

一个例子（部分完成的候选系统矩阵）在图 10-7 中列出了 5 个候选方案中的 3 个。在图 10-7 中，矩阵用来提供概要特征，内容涉及系统被计算机化的部分、企业收益和需要的软件工具和/或应用。除了在一个或两个单元格上的输入有所不同以外，矩阵中两列之间可能是类似的。如果我们正考虑 3 个以上的候选方案，可以使用多页形式。可以复制一个字处理软件“表格”模板来创建候选系统矩阵。

特 征	候选方案 1	候选方案 2	候选方案 3	候选方案……
计算机处理部分 简单描述在这个候选方案中系统将被计算机处理的部分	将购买 Entertainment Software Solutions 公司的 COTS 软件包 Platinum Plus，并定制以满足“会员服务系统”需要的功能	“会员服务系统”与同订单履行有关的仓库职能	同候选方案 2	
收益 简单描述这个候选方案将实现的业务收益	这个方案可以被快速地实现，因为它是一个购买的方案	充分地支持用户需要的 SoundStage 公司的业务过程，并且与会员账号更有效的交互	同候选方案 2	
服务器和工作站 描述支持这个候选方案需要的服务器和工作站	技术架构规定使用 Pentium III、MS Windows 2000 类服务器和工作站（客户端）	同候选方案 1	同候选方案 1	
需要的软件工具 设计和构建这个候选方案需要的软件工具（例如，数据库管理系统、模拟器、操作系统、语言等）。如果要购买应用软件包，则这一条一般无意义	用来定制软件包的 MS Visual C++ 和 MS Access，以提供报告编写和集成	MS Visual Basic 5.0 System Architect 2001 Internet Explorer	MS Visual Basic 5.0 System Architect 2001 Internet Explorer	

图 10-7 候选系统矩阵示例

特 征	候选方案 1	候选方案 2	候选方案 3	候选方案……
应用软件 描述要购买、构建和评估的软件	软件包方案	定制方案	同候选方案 2	
数据处理方法 通常是以下的某些组合：联机、批处理、延期批处理、远程批处理和实时	客户/服务器	同候选方案 1	同候选方案 1	
输出设备和建议 描述要使用的输出设备、特殊的输出需求（例如，网络、预打印的表格等）和输出因素（例如定时约束）	(2) HP4MV 部门激光打印机 (2) HP5SI LAN 激光打印机	(2) HP4MV 部门激光打印机 (2) HP5SI LAN 激光打印机 (1) PRINTRONIX 条形码打印机（包括软件和驱动程序） 必须设计适应 VGA 分辨率的 Web 页面。所有的内部屏幕将按照 SVGA 分辨率设计	同候选方案 2	
输入设备和建议 描述要使用的输入方法、输入设备（例如，键盘、鼠标等）、特殊的输入需求（例如，输入数据的新表格或者改进的表格）和输入因素（例如，实际输入的定时要求）	键盘和鼠标	Apple “Quick Take” 数码相机和软件 (15) PSC Quickscan 激光条形码扫描仪 (1) HP Scanjet 4C 平板扫描仪键盘和鼠标	同候选方案 2	
存储设备和建议 简单描述将存储什么数据，将从现有存储访问什么数据，将使用什么存储介质，将需要多少存储空间，以及将如何组织数据	MS SQL Server DBMS, 100GB 阵列存储功能	同候选方案 1	同候选方案 1	

图 10-7 （续）

10.4.2 可行性分析矩阵

第二个矩阵使用候选系统的分析和等级评定补充候选系统矩阵。称为可行性分析矩阵。

矩阵的列与候选系统矩阵相同的候选方案对应，有些行对应本章介绍的可行性准则，但增加了几行描述候选系统的一般方案和等级评定。一般格式如图 10-8 所示。

	权 重	候选方案 1	候选方案 2	候选方案 3
描述				
运行可行性				
文化可行性				
技术可行性				
经济可行性				
进度可行性				
法律可行性				
加权得分				

图 10-8 可行性分析矩阵模板

单元格包含了对每个候选方案的可行性评估注解。每行可以为每个准则指定一个等级或者积分（例如，对于运行可行性，候选方案可以分级为 1、2、3 等）。当对每个候选方案在每个准则上分级并打分后，最终的分级或打分记录在最后一行。小心，不是所有的可行性准则都同等重要。在分配最后的等级评分之前，你可以快速地去掉被某些准则认为不可行的候选方案。在现实中，这种情况不会经常发生。

图 10-9 显示了一个完整的可行性分析矩阵。在图中，为每个候选方案提供了可行性评估。在这个例子中，分值直接记录在单元格中作为每个候选方案的可行性评价准则的评估结果。权重是可以进行量化分析的。但要记住，在任何一个评价准则方面完全不可行的方案要删除掉。例如，只有通过破坏与供应商的合同才能实现的方案不应被考虑。

	权重	候选方案 1	候选方案 2	候选方案 3
描述		为会员服务部购买商业现货软件包	使用标准的 VB.NET 和 SQL Server 数据库内部编写新的应用软件	使用 PowerBuilder 重写现有的内部应用软件
运行可行性	15%	仅支持会员服务部的需求，而且当前的业务过程将不得不被修改以发挥软件功能优势。另外，还存在系统安全问题 得分：60	完全地支持用户需求的功能 得分：100	完全地支持用户需求的功能 得分：100
技术可行性	20%	Platinum Plus 软件包的当前发行版是版本 1.0，并刚刚上市 6 个星期。评估产品的成熟度有风险，而且公司需要为技术支持支付额外的月租 需要雇用或培训 Java J2EE 专业人员来为集成需求进行修改 得分：50	该方案需要用 VB.NET 编写程序。尽管当前的技术人员仅有 PowerBuilder 经验，但是找到有经验的 VB.NET 程序员要相对容易些 得分：95	尽管当前的技术人员熟悉 PowerBuilder，但管理层担心 Sybase 公司的 PowerBuilder 的前途。MS SQL Server 是目前公司的数据库标准，并同 Sybase DBMS 竞争。因此，我们不能保证 Power Builder 的未来版本将与目前版本的 SQL Server “工作良好” 得分：60
经济可行性 开发费用： 回报期（折扣的）： 净现值： 详细计算：	30%	约 350 000 美元 约 4.5 年 约 210 000 美元 见附件 A 得分：60	约 418 000 美元 约 3.5 年 约 307 000 美元 见附件 A 得分：85	约 400 000 美元 约 3.3 年 约 325 000 美元 见附件 A 得分：90
进度可行性	10%	少于 3 个月 得分：95	9 ~ 12 个月 得分：80	9 个月 得分：85
加权得分	100%	60.5	92	83.5

图 10-9 可行性分析矩阵示例

10.5 系统方案建议

回顾在第 4 章中学习的内容，决策分析阶段包括确定候选方案，分析并比较那些方案，然后选择整体最佳的方案，最后推荐一种方案。我们刚刚学习了如何完成前三个任务，现在学习如何推荐一种方案。

推荐一种方案包括生成一种系统方案建议，这个交付成果通常是面向系统所有者和用户的一份正

式书面报告或者口头汇报。所以，系统分析员应该能够在不陷入技术性问题的情况下撰写一份正式的业务报告，并进行一次业务演示汇报。下面概述有关书面报告和演示汇报的一些重要概念。

10.5.1 书面报告

书面报告是分析员与系统用户沟通中用得最多的一种方法。我们往往会编写篇幅巨大的多卷报告，看上去十分惊人。有时这样一个报告是需要的，但不是经常需要的。如果你把一份 300 页的技术报告放在一位经理的桌子上，你可以想到那个经理只可能浏览它而不可能阅读它——可以确信报告不会得到仔细的研究。

10.5.1.1 书面报告的长度

在经历教训和过失之后，我们已经学会了使用下面这些一般性指南来控制报告长度：

- 对高级管理人员——1~2 页
- 对中级管理人员——3~5 页
- 对主管级管理人员——少于 10 页
- 对办事员级人员——少于 50 页

可以组织一个由面向不同级别管理人员的子报告构成的大报告。这些子报告通常作为报告的前面几节，而且通常作为这个报告的总结，并专注于底线问题。

10.5.1.2 书面报告的组织

有几种通用的模式可以用来组织报告。每个报告都包括主要部分和次要部分。主要部分表示报告期望说明的实际信息，例如，引言和结论。

虽然主要部分表示了实际的信息，但是所有的报告都还包括次要部分。次要部分封装报告以便读者可以容易地识别报告及其主要部分。次要部分还给报告增加了专业修饰。

如图 10-10 所示，主要部分可以用两种格式来组织：事实型格式和管理型格式。事实型格式是很传统的格式，最适合于对事实、细节以及结论感兴趣的读者，这是我们将用来向系统用户说明详细需求和设计说明的格式。但事实型格式不适合大多数管理人员。

事实型格式	管理型格式
I 引言	I 引言
II 方法和程序	II 结论和建议
III 事实和细节	III 事实和细节的总结与讨论
IV 事实和细节的讨论与分析	IV 方法和程序
V 建议	V 最终结论
VI 结论	VI 含事实和细节的附录

图 10-10 书面报告的格式

管理型格式是一种现代的面向结果的格式，许多管理人员喜欢采用这种格式。

该格式是为那些对结果而非事实感兴趣的读者设计的，它首先介绍结论或建议。任何读者可以直接阅读这份报告，直到报告的详细程度超出了他们的兴趣范围。

两种格式都包括了一些公共的元素。引言应该包括 4 个组件：报告的目的、问题陈述、项目范围和报告内容的叙述性解释。方法和程序节应该简要地解释报告中包含的信息是如何得到的。例如，研究是如何进行的或者新系统将如何设计。报告的主体将是事实节。应该命名这一节以描述被表现的事实数据的类型（例如，“现有系统描述”、“替代方案的分析”或者“设计说明”）。结论应该简要地总结报告，验证问题陈述、调查结果和建议。

图 10-11 显示了报告的次要（或封装）部分，以及它们同主要部分的关系。这些要素中有许多是自我解释的，我们这里只简要地讨论那些不能自我解释的要素。如果没有转送函，报告不应该发出去，转送函应该是清楚可见的，而不应该放在报告的封面里面。转送函陈述了报告需要哪类行动，也可以提醒读者注意项目中应受到特别注意的特征。而且，转送函是致谢的合适地方。

转送函
标题页
目录
图、示例和表格清单
摘要或总结
(主要部分——事实型格式或者管理型格式的 报告主体——在报告的这个部分表现)
附录

图 10-11 一个书面报告的次要部分

摘要或总结以 1~2 页篇幅总结整个报告。摘要既有助于读者判断报告是否包含了他们需要知道的信息，也可以作为最高层的总结报告。几乎每个管理人员都会阅读这些总结，大多数管理人员将继续阅读后续内容，但可能跳过详细的事实和附录。

10.5.1.3 撰写报告

图 10-12 显示了撰写一个正式报告的正确程序，下面是一些要遵循的指南：

- 段落应该传达单一思想。段落之间应该流畅，一个接一个。段落结构差一般均可以归咎到提纲的缺陷。
- 语句不应该太复杂。平均语句长度不应该超过 20 个字，研究表明，超过 20 个字的句子难以阅读也难以理解。
- 用主动语气写。如果一直使用被动语气，就会显得冗长而且令人厌烦。
- 消除行话、大话和多余的话。例如，用“数据库管理系统”代替“DBMS”，用“那么”代替“相应地”，试着使用“有用的”代替“有利的”，使用“清楚地”代替“它是明显的”。

请购买一本 William Strunk, Jr. 和 E. B. White 合著的《The Elements of Style》。这个经典的平装本可能是最有价值的书之一，虽然它只是一本几乎只比口袋大一点的书，但它却是一个信息金矿。

10.5.2 正式汇报

为了与系统开发项目涉及的不同人沟通信息，系统分析员经常需要做一个正式汇报。正式汇报是用来兜售新想法并获得新系统认可的专门会议。还可以用于以下目的：兜售一个新系统、兜售新想法、兜售变化、阻止批评、讨论担心的问题、验证结论、澄清事实和报告进展。在许多情况下，正式汇报可以补充更详细的书面报告。

有效且成功的汇报需要做大量的准备工作。分配给汇报的时间经常是短暂的，所以汇报的组织 and 格式十分关键。你不可能临时准备并期望获得接受。

汇报具有施加影响的优点，这是通过立即反馈和自发地响应实现的。听众能够响应汇报人，汇报人可以使用强调、定时暂停和肢体语言来传达书面语言所不能传达的消息。汇报的缺点是汇报的材料容易被忘记，因为内容是说出来的，而且视觉表现又是短暂的。所以汇报经常后附一份书面报告，要么是总结报告，要么是详细报告。

10.5.2.1 准备正式汇报

了解你的听众特别重要，当你的汇报是要兜售新想法和新系统时，这一点就更重要了。系统分析员经常被看作是一个组织中可怕的变革因子，正如 Machiavelli 在他的经典著作 *The Prince* 中写的那样。

没有什么比贯彻一个新秩序更难以实施，或者处理起来更危险的了。因为改革者有敌人，这些敌人从旧秩序中获益；而改革者又只是有保留的支持者，虽然支持者都将从新秩序中获益，但这种有保留部分地来自对敌人的恐惧，部分则来自人类的怀疑天性。人们不相信任何新事物，除非他们已经有了实际的经验。⊖

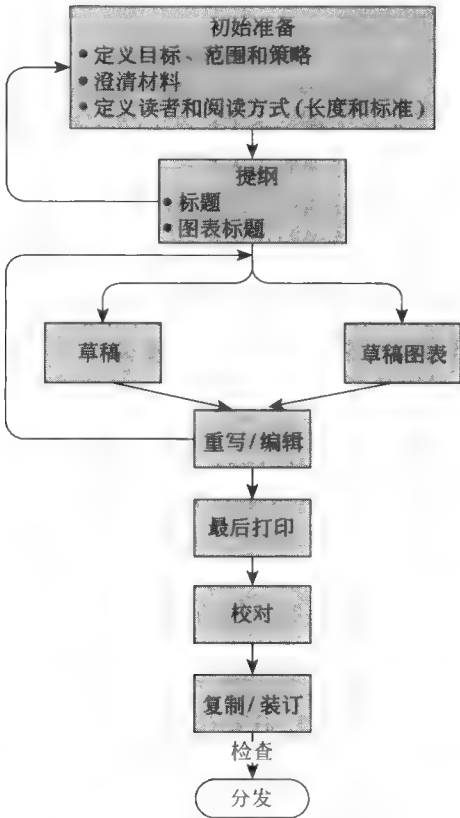


图 10-12 撰写一个报告的步骤

资料来源：Keith London 版权。

⊖ Niccolo Machiavelli, *The Prince and Discourses*, trans. Luigi Ricci (New York: Random House, 1940, 1950). Reprinted by permission of Oxford University Press.

人们倾向于反对变化，目前熟悉的方式给人以安逸的感觉。然而分析员最重要的工作就是带来变化（在方法、程序、技术等方面）。一个成功的分析员必须是一个优秀的销售员。对一个分析员来说，正式地研究销售学是完全合适的（极力推荐）。为了有效地表现和兜售变化，你必须相信自己的想法，并且用事实支持它们。再次重申，准备是成功的关键！

首先，确定你对汇报的期望。例如，你寻求批准继续该项目，你试图证实事实，等等。一个汇报总结了预期的想法和建议。

主管通常对过分的细节感到厌倦。为了避免这一点，你的汇报应该小心地围绕分配的时间进行组织（通常 30 ~ 60 分钟）。尽管每个汇报都不同，但是你可以尝试图 10-13 中的组织和时间分配建议。该图说明了口头汇报的一些典型主题，以及那些主题允许的时间总量。注意这个专门的提纲是用于系统分析汇报的，其他类型的汇报可能稍有不同。

还能做什么来准备这个汇报？因为汇报时间有限，所以应该使用可视化辅助工具（绘制的翻转图、高射投影仪、PowerPoint 演示幻灯片等）来支持你。正像一个书写的段落，每个可视化辅助应该传达一个想法。当准备图片或文字时，请使用图 10-14 中显示的指南。

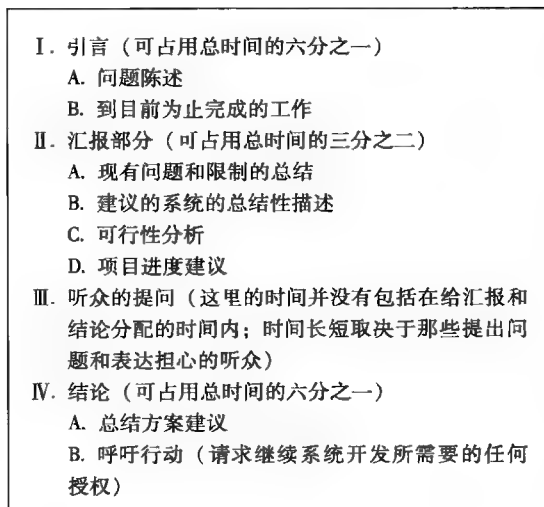


图 10-13 一个口头汇报的典型提纲和时间分配



图 10-14 可视化辅助的指南

资料来源：Keith London 版权。

微软的 PowerPoint 包含了称为向导的软件指南，用来辅助新手创建具有专业表现力的汇报。它分步指导用户工作，询问一系列的问题并根据回答裁剪汇报。为了吸引听众的注意力，在汇报开始时可以分发可视化辅助的副本。这样，听众就不用记那么多笔记。

最后，在你可以找到的最重要听众面前预先汇报一次。就像一次正式汇报那样地做一次汇报，然后让他们提出批评和建议，并试着回答这些问题。

10.5.2.2 进行正式汇报

下面一些额外的指南可以改进实际汇报的效果：

- 专业地着装。你着装的方式将会对人产生影响，John T. Malloy 的 *Dress for Success* 和 *The Woman's Dress for Success Book* 两本书，既是有关衣着忠告方面的极好的书，也是关于着装对管理层的影响的研究结果的好书。
- 当进行汇报时，避免使用“我”。使用“你们”和“我们”向管理层指出建议系统的所有权关系。

- 保持同参加人员的目光接触并保持信任的气氛。如果你没有向管理层显示出对自己的建议的自信,那么管理层为什么应该相信你的建议?
- 了解你自己的特殊习惯。一些最常见的特殊习惯包括使用太多的手势、踱步和反复说“你们知道”或者“是的”。尽管特殊习惯本身并不会与要传达的信息冲突,但它们可能会转移听众的注意力。

有时当你做汇报时,听众中有些成员可能并没有在听。注意力分散可能有几种形式:有些人可能投入到争论中,有些人可能在走神,有些人可能忙着看表,有些在听的人可能有迷惑的表情,而有些人则可能毫无表情。以下的建议可能对保持人们的注意力有用:

- 停止说话。安静也可以是震耳欲聋的。最好的公共演说家知道如何使用动态的停顿表示特别的强调。
- 提问一个问题,然后让听众回答。这使听众参与到汇报中,并且对于停止争论十分有效。
- 讲一个小幽默。你不必是一个天才戏剧演员,但每个人都喜欢笑。讲一个关于你自己的笑话。
- 使用一些道具。使用某种类型的可视化辅助工具可以使你的观点清晰。在黑板上画图,或在你的笔记本背面演示,创建一个物理模型使信息更容易理解。
- 改变你的声调。通过声音的提高或柔和,强迫听众听得更仔细,并使他们更容易听到。任何一种方式都行,只要你对听众习惯的方式有所变化,那就是获得并抓住注意力的最好方式。
- 做一些完全意料之外的事。放下书,撕你的笔记本,摇动你的钥匙。做一些意料之外的事几乎总是抓住注意力的好办法。

通常一个正式的汇报将包括留给听众提问的时间。这个时间很重要,因为它使得你可以澄清任何不清楚的要点,并进一步强调重要的思想,也使观众可以与你进行交流。但是,有时在汇报后回答问题可能是困难的,而且是易受挫折的。当回答问题时,我们建议以下指南:

- 总是严正地回答问题,哪怕你认为那是一个愚蠢的问题。记住,如果你让某人觉得因为问了一个“愚蠢”的问题而感到很傻,你就冒犯了那个人。而且,听众中其他人将不会再提问题,因为害怕受到同样的对待。
- 向提出问题的人和全体听众回答。如果你把所有的注意力导向提问的人,其他观众将会感到厌烦;如果你没有给提问的人以足够的注意,那个人就会不满意。重要的是要达到一种平衡。如果问题不是观众普遍感兴趣的问题,可以事后专门向那个人回答。
- 总结你的回答。回答问题要充分,但又不要陷入细节。
- 限制你花在回答任何一个问题上的时间。如果需要额外的时间来回答问题,就等到汇报结束之后再继续回答。
- 要诚实。如果你不知道问题的答案,就承认它。不要试图不懂装懂,听众最终将会发现你不懂,这将有损你的信誉。相反,应该承诺找出问题答案并在以后进行报告,或者要求听众中的某人做一些研究并在以后汇报结果。

10.5.2.3 正式汇报的后续工作

如前所述,正式汇报的后续工作特别重要。因为用于汇报的口头言语和感人的可视化辅助工具并不总是留下持续的印象,所以大多数汇报之后都会呈上书面报告,这些书面报告给听众提供了更持久的信息。

复习题

1. 逐步投入法对于可行性分析有什么意义?
2. 在开发周期中有哪些可行性分析检查点?在每个检查点应该做什么?
3. 运行可行性准则的目的是什么?
4. 为什么找出最终用户和管理人员对系统分析员提出的问题解决方案的感受十分重要?
5. 什么时候进行可用性分析?可用性分析的目的是什么?
6. 技术可行性准则的目的是什么?
7. 开发费用和运行费用各有什么特点?每种费用各举3个例子。
8. 列举有形收益的5个例子。

- 9. 当评估经济可行性时，为什么资金的时间价值概念是一个基本的考虑？
- 10. 决定一个项目的成本效益的最常用的技术是什么？
- 11. 候选系统矩阵和可行性分析矩阵的用途是什么？

问题和练习

- 1. 本书描述了一种用于可行性分析的逐步投入法。
 - a. 解释这种方法，指出为什么本书推荐这种方法。
 - b. 由于发生了什么变化或者事件使得这种方法是可取的？
 - c. 如果项目变得不可行，企业应该取消项目吗？
- 2. 本书描述了度量可行性的三个检查点。
 - a. 这三个检查点是什么？
 - b. 通常，在每个检查点决定可行性的正确性有多大？
 - c. 哪个检查点是最关键的检查点？
- 3. 有哪 6 类可行性准则？每一类用于评估可行性的标准是什么？
- 4. 你是一个项目的系统设计人员，该项目即将完成系统设计阶段。已经开发了一个可工作的原型系统，你的任务是进行可用性分析。起草一份一至两页纸的计划，详细描述你进行可用性分析的方法。
- 5. 你是一个在拥有 300 名雇员的中型企业的 IT 商店工作的系统分析员。该企业正处于一个项目的系统设计阶段，为所有的雇员开发一个电子活动报告系统，代替当前的手工方法。所有的工作都是在室内进行的，除了几个提供辅助服务的顾问，例如 IV&V。该系统将使用雇员现有的台式机，虽然需要增加几台专用服务器。用户界面很直观，但项目要求对所有雇员进行半天的培训，学习使用新系统的政策和程序。该系统没有使用任何新技术，IT 技术部门拥有大量的专家。创建一份工作表，详细描述估计的一次性开发费用和持续性的运行费用。顺便说一下，在你的企业中，系统分析员的平均工资和收益是 40 美元/小时，你可以以这个数据作为估计项目中涉及的其他人员的工资和收益的基础。
- 6. 在题 5 的项目中，注意到电子活动报告系统将代替当前的手工系统。描述可预期的有形收益。采用“最佳猜测”方法，计算企业的年度节省费用。用计算解释你的假设。
- 7. 你正在设计一个基于 Web 的系统，通过该系统你的地区办公室可以通过联机提交他们的销售报告，而不用再手工填写然后邮寄了。已经确定了三个候选方案。它们的估计生命期收益和估计生

- 12. 对于书面报告，事实型格式和管理型格式之间有什么区别？
- 13. 用什么步骤撰写报告？
- 14. 汇报演示有什么优缺点？
- 15. 进行正式汇报应该遵循什么原则？

命期费用在下面显示。所有的值都已经按照项目的 5 年生命期进行了时间调整。

	估计生命期收益	估计生命期费用
候选方案 1	640 000 美元	172 000 美元
候选方案 2	640 000 美元	160 000 美元
候选方案 3	640 000 美元	185 000 美元

- 按照投资回报分析，哪个候选方案提供了最高的 ROI？如果企业设置最小生命期 ROI 为 80%，这些方案中哪一个经济上最可行？
- 8. 确定候选方案有哪些不同的技术或方法？如果你不得不只选择其中的一种方法，你会选择哪一个，为什么？
 - 9. 你是一个生产压缩机使用的重型动力工具的公司的系统设计人员。每月你的地区销售和服务中心将纸质的保修修理订单打包，发送给总部，在总部运行一个遗留的大型机批处理系统。然后生成一份报告，工程师用报告分析在新的模型中任何问题趋势的信号。公司的 CEO 认为这个过程在如今高度竞争的商业环境中太慢了，想尽快用更现代的系统替换掉这个遗留系统。确定至少三个候选方案，用候选方案矩阵描述它们，使用图 10-7 为例。
 - 10. 准备一个可行性分析矩阵，使用你在题 9 中确定和描述的候选方案。使用图 10-9 作为模板，但选择你认为这种情况下最合适的权重因子。用于练习的目的，你可以提供一个经济可行性的估计。
 - 11. 一旦完成了可行性分析矩阵，就要撰写可行性报告。对于本练习，给执行管理层准备一份可行性报告，使用图 10-10 中显示的格式。
 - 12. 你被要求在周例会上给每个部门的执行经理汇报可行性分析和建议。准备一组 PowerPoint 幻灯片，用作汇报时的视频辅助。
 - 13. 如果你想让你的汇报具有信息含量、有说服力并受到欢迎，说出至少 10 件你不应该做的事情。

项目和研究

1. Steve McConnell 编写了许多关于软件工程和开发的书籍。在他的 *Rapid Development* 中, McConnell 指出, 在工程中相比实际的构造工作, 设计工作通常只是整个项目很小的一部分。他对大桥建设项目进行了比较, 其中设计占 10%, 构造占 90%。对于软件开发项目, 其中设计通常至少占整个项目工作的 50%。研究软件工程相比其他类型的工程这一特殊的差别问题, 用一份一至两页纸的论文总结你的分析和结论。其他的软件工程界领袖对这个问题有什么看法吗?
2. 你是你们州的高速公路巡逻队总部的一名系统分析员, 巡逻队在整個州都有现场办公室。目前, 交通事故报告是由高速公路巡逻员在现场手写的, 由他们的警官检查、临时存储, 然后每月批处理地发送到总部。每一份都通过数据操作员输入一个老式的大型主机系统。来自每个乡村的巡逻办公室的报告都输入后, 计算机操作员运行使用 JCL 的编辑程序。

有严重错误或漏项的报告被拒绝, 并退回原先的乡村高速公路巡逻办公室进行改正。当为所有的乡村完成了编辑之后, 就运行一个修改程序将月份批处理的交通事件报告添加到报告主文件。按每个季度和每年生成统计报告。从收到一批报告到修改完主文件的整个过程一般要花三个月的时间。管理层想用一个新系统代替旧系统, 新系统具有更现代化、更少劳力、更准确、用户更容易访问并且能减少准备统计报告的周期时间的优点。作为项目团队的成员, 你的任务是准备可行性研究报告 (FSR)。

- a. 你认为应该考虑哪些选项或者候选方案? (除了“什么也不做”或者“维持现状”至少提出三个候选)
- b. 准备一个候选方案矩阵描述每个方案的特征, 使用图 10-6 中的候选方案矩阵模板。
- c. 扩展候选方案矩阵, 使用图 10-7 的模板。
- d. 评估每个方案的运行可行性、技术可行性和进度可行性, 使用本书描述的技术和图 10-9

的模板。

3. 基于问题 2 描述的情景:
 - a. 为每个候选方案准备一份估计费用工作表, 使用图 10-2 的模板。
 - b. 评估每个候选方案的经济可行性, 使用本书描述的三种技术之一。你使用了哪种技术, 为什么?
 - c. 将经济可行性分析增加到上一问题的可行性分析矩阵中。
 - d. 比较每个候选方案并打分。每个可行性评价指标使用与本书中不同的权重因子。
 - e. 在你的可行性分析矩阵中你为不同的可行性评价指标选择了什么权重因子? 为什么?
4. 管理层对你在可行性分析矩阵上的出色工作印象深刻, 并且要求你准备系统建议报告。撰写一份系统建议, 其主要读者是中层业务和 IT 经理, 但也会被执行主管和首席信息官阅读。使用图 10-10 中显示的恰当的格式。
5. 中层管理人员对你的系统建议很感兴趣。他们现在想让你给部门的高层领导准备并进行一次正式汇报。
 - a. 描述你为准备正式报告应该进行的步骤。
 - b. 准备一份 PowerPoint 汇报片, 使用本书建议的或者其他的书和文章中关于 PowerPoint 汇报片指南。
 - c. 在准备正式汇报中, 根据你的考虑, 哪些是应知道的最关键的事情? 为什么?
6. 准备系统建议和可行性研究报告存在很多不同的格式、模板和方法。搜索网络看看你可以找到其他的什么工具和技术。
 - a. 描述你找到的格式及其来源。
 - b. 对比你找到的各种格式, 并同本书中的格式比较。它们之间有什么区别吗?
 - c. 你认为有一种格式明显优于其他格式吗? 如果有, 请描述这个格式, 为什么你觉得这个格式较好。
 - d. 为你的企业创建你认为是最理想的 FSR 模板。

小型案例

上一章中的杂货店 Wow Munchies 正考虑为购买食品的顾客开发一个联机网站。商店所有者相信这个功能将使商店能够从附近的 Fast Food 公司抢占市场份额, 该公司有一个网站并给顾客分发食品。这个网站允许顾客购买商店中上架的任何东西。商店

将不送食品, 但将把食品包装好, 等着顾客在指定的时候来拿。Wow Munchies 只有一个前台。

1. 进行一个运行可行性研究。你认为网站能使 Wow Munchies 公司如它所设想的那样获取市场份额吗? 什么因素将会影响到这个网站的运行成功?

- 提交你的论文、支持文档、表格和你所进行的任何交谈记录。
2. 进行一个技术可行性研究。你推荐这个公司使用什么技术创建和维护他们的网站（例如：语言、专门的主机、加密）？为什么你的选择会影响到站点的可行性？提交你的论文、支持文档、表格和你进行的任何交谈记录。
 3. 为投资电子商务网站进行一个经济可行性研究。你使用多少贴现率？为什么？提交你的论文、支持文档、表格和你进行的任何交谈记录。
 4. 开发一个完成这个网站的进度可行性研究。你看到任何会引起进度延迟或期限超期的因素了吗？提交一篇短文和一张甘特图。

团队和个人练习

1. 圆桌讨论：ROI 分析经常考虑 3 ~ 5 年的技术“生命期”。你认为在那个时间周期以后技术对业务还有影响吗？为什么？
2. 个人：在上一章中，你讨论了（圆桌讨论形式）经济成功中知识和信息的重要性。每年大学助学金都有未发出的，因为学生没有申请（可能因为他们不知道助学金的存在）。调研你可以得到的大学助学金，至少申请一个。
3. 团队/班级讨论：拥有成功的信心有什么意义？你认为人们相信自己的能力会影响他们实际的成功吗？在另一方面，你认为人们缺少自信会影响他们成功的能力吗？你可以做什么来进一步提高你成功的信心？

系统设计方法

第三部分介绍系统设计方法。第 11 章（系统设计）通过介绍系统设计活动为所有后续章节提供基础。系统设计包括准备基于计算机的详细规格说明，这个规格说明实现了系统分析和系统原型构造期间说明的需求。至于信息系统开发，系统设计包括配置、获取和设计集成阶段。

第 12 章（应用架构和建模）介绍物理过程和数据设计。专门介绍有关共享的数据和过程如何分布的设计决策，这个设计决策得出一个由设计单元构成的应用架构，设计单元可以分配给不同团队成员进行详细设计、构造和单元测试。

第 13 章（数据库设计）介绍从第 7 章开发的数据模型到物理数据存储的设计。

第 14 章（输出设计和原型化）讲授输出设计和原型化技术，介绍了输出的不同类型、格式和介质，讨论了最常用的图形类型。该章演示了如何打印设计和原型以及显示输出。

第 15 章（输入设计和原型化）讲授输入设计和原型化技术，重点介绍输入的格式、方法、介质、人的因素和内部控制，讨论了在图形用户界面（GUI）屏幕设计中，如何正确使用基于屏幕的数据输入控制。该章还重点介绍了原型化技术，作为一种发现、记录和沟通输入设计需求的方法。

第 16 章（用户界面设计）讲授用户界面设计和原型化技术。你将学到如何为一个应用系统开发一个既友好又有效的界面。用户界面的设计是至关重要的，因为用户对系统的接受程度经常取决于一个友好且易用的界面。用于获得输入和输出的 GUI 界面设计在第 14 章和第 15 章中演示。

最后，第 17 章（使用 UML 进行面向对象设计和建模）介绍采用系统开发的面向对象方法进行系统设计的工具和技术。

系统设计

本章概述和学习目标

在本章中，你将学到更多有关系统开发设计阶段的知识，具体将学到以下内容：

- 按照信息构件描述设计阶段。
- 确定并区分几个系统设计策略。
- 描述一个内部开发项目的计算机系统方案中设计阶段的任务。
- 描述一个包含采购商业系统软件的计算机系统方案中设计阶段的任务。

虽然本章介绍了一些系统设计技术，但讲授这些系统设计技术并不是本章的目的。本章仅仅讲授系统设计的过程，并向读者介绍一些将在后续章节中讲授的技术。

本章关键术语

模型驱动设计（model-driven design）是一种系统设计方法，强调通过绘制系统模型记录系统的技术或实现方面。

现代结构化设计（modern structured design）是一种系统设计技术，它将系统过程分解成可管理的构件。

报价申报书（request for quotation, RFQ）是一份正式的文档，为一个供应商提供对某个应用软件包的业务、技术和支持需求交流的文档，该供应商已经被确定能够提供该软件包和服务。

建议申报书（request for proposal, RFP）是一份正式的文档，给几个供应商交流对一个应用软件包的业务、技术和支持需求，这些供应商可能希望竞争该软件包和服务的销售。

11.1 什么是系统设计

第2章介绍了系统开发过程，在该章中，我们有意地将讨论限制在仅仅简要地介绍每个开发阶段。在本章中，我们将更深入地讲解系统分析之后的系统设计阶段。信息系统设计被定义为那些用来说明一个详细的计算机系统方案的任务，也称为物理设计。因此，系统分析强调了业务问题，系统设计则专注于系统的技术性 or 实现方面。

系统设计由“系统设计人员”的技术性关注所驱动。因此，它涉及从“系统设计人员”角度看待的信息系统构件。“系统分析员”是系统设计的推动者。

我们中大多数人对设计过程的定义过于局限：我们想象自己绘制计算机系统的蓝图，然后由自己或者程序员编程和开发，所以我们要设计输入、输出、文件、数据库和其他计算机组件。实际上，许多公司购买的软件比他们内部编写的多，对此你不应该感到吃惊——为什么要重新去做那些已经现成的东西呢？许多系统都很通用，计算机供应商已经编写了很多软件包，这些软件包可以买到，并能够被修改，以实现用户的需求。

本章既从内部开发（或“构造”）项目的角度，也从软件采购（或“购买”）项目的角度介绍系统设计。下面首先介绍系统设计的一些整体策略。

11.2 系统设计方法

有许多策略或技术可以用来进行系统设计，如：现代结构化设计、信息工程、原型化、JAD、RAD和面向对象设计。这些策略经常被当作相互竞争的替代方法，但实际上，它们可以互相补充。下面简要地介绍这些策略，及其适合的项目范围和目标。我们的目的仅仅是建立一个高层概念，后续章节将具体讲授这些技术。

注意 第2章中为某些方法定义了方法学“开发路线”。

11.2.1 模型驱动方法

结构化设计、信息工程和面向对象设计都是模型驱动方法的例子。**模型驱动设计**强调通过绘制图形化系统模型描述新系统的技术或实现方面。

设计模型通常从前面的（在第4章讨论的）模型驱动分析中开发的逻辑模型导出。最终，系统设计模型将作为构造和实现新系统的蓝图。

如今，模型驱动方法几乎总要使用自动化工具。有些设计人员使用通用绘图软件绘制系统模型，例如 Visio Professional 或 Corel Flow；其他设计人员和组织要求使用基于资料库的 CASE 工具或建模工具，例如 System Architect、Microsoft Visio、Visible Analyst 或者 IBM 的 Rational。CASE 工具提供了一致性和完整性检查，以及基于规则的错误检查。

下面将简要地介绍最常见的模型驱动设计方法，它们是第2章中介绍的模型驱动方法学和开发路线的特征化。

11.2.1.1 现代结构化设计

结构化设计技术有助于开发人员处理程序的规模和复杂性。**现代结构化设计**是一种面向过程的技术，用于将一个大的程序分解成一个容易实现和维护（修改）的计算机程序模块层次。同义词（尽管技术上不准确）是自顶向下程序设计和结构化编程。

这个概念很简单——把一个程序设计成一个自顶向下的模块层次。一个模块就是一组指令——一个程序段、程序块、子程序或子开发路线，这些模块的自顶向下结构按照各种设计规则和设计指南进行开发（因此仅仅绘制一个程序的层次图或结构图并不是结构化设计）。

结构化设计被认为是面向过程的技术，因为它强调的是信息系统中的“过程”构件——特别是软件过程。结构化设计力图将一个程序分解成具有如下特征的自顶向下模块层次：

- 模块应该是高度**内聚**的；也就是说，每个模块应该实现一个功能，而且仅实现一个功能。这使得模块在未来的程序中可复用。
- 模块应该是松散**耦合**的；换句话说，模块间的相互依赖应该最小。这使将来一个模块的变化对另一个模块造成的影响达到最小化。

正如将在第17章中讲述的，内聚和耦合也是面向对象世界中的重要概念。从结构化设计中得到的软件模型称为**结构图**（见图11-1），它由通过程序的数据流导出。结构化设计在系统设计期间进行，但它并没有涉及设计的所有方面。例如，结构化设计将不能进行设计输入、输出或数据库。

如今结构化设计在许多应用系统中已经不再那么流行，这是因为那些应用系统要求使用更新的事件驱动和面向对象编程技术。当涉及到基于大型主机应用软件设计时，结构化设计仍是一种流行的技术，并常用于解决“系统”级的耦合和内聚问题。

11.2.1.2 信息工程

如第4章所述，信息工程（Information Engineering, IE）是一种模型驱动的、以数据为中心的、对过程敏感的技术，用于计划、分析和设计信息系统。IE的主要工具是数据模型图（见图11-2）。IE要求进行企业级的需求分析，从这个需求分析中导出信息系统应用，并对它们排列优先次序。在IE中确定的应用成为进行系统开发的项目，其他各种系统分析和设计方法可以用于这些项目中，这些方法可能包括多种技术的组合，例如，现代结构化分析（在第4章讨论）、现代结构化设计、原型化以及面向对象分析和设计。

11.2.1.3 原型化

传统的物理设计是一种使用纸和笔的工作过程，分析员绘制描述了输出、输入和数据库的布局或者结构以及对话和程序流程的图形。这是一个耗时的过程，而且容易出现错误和遗漏，最终的书面说明经常不充分、不完整或者不准确。

如今，许多分析员和设计人员更喜欢使用原型化技术。这是一种现代的工程设计方法。原型化方法是一种反复过程，它需要设计人员和用户之间保持紧密的工作关系。这个方法有以下几个优点：

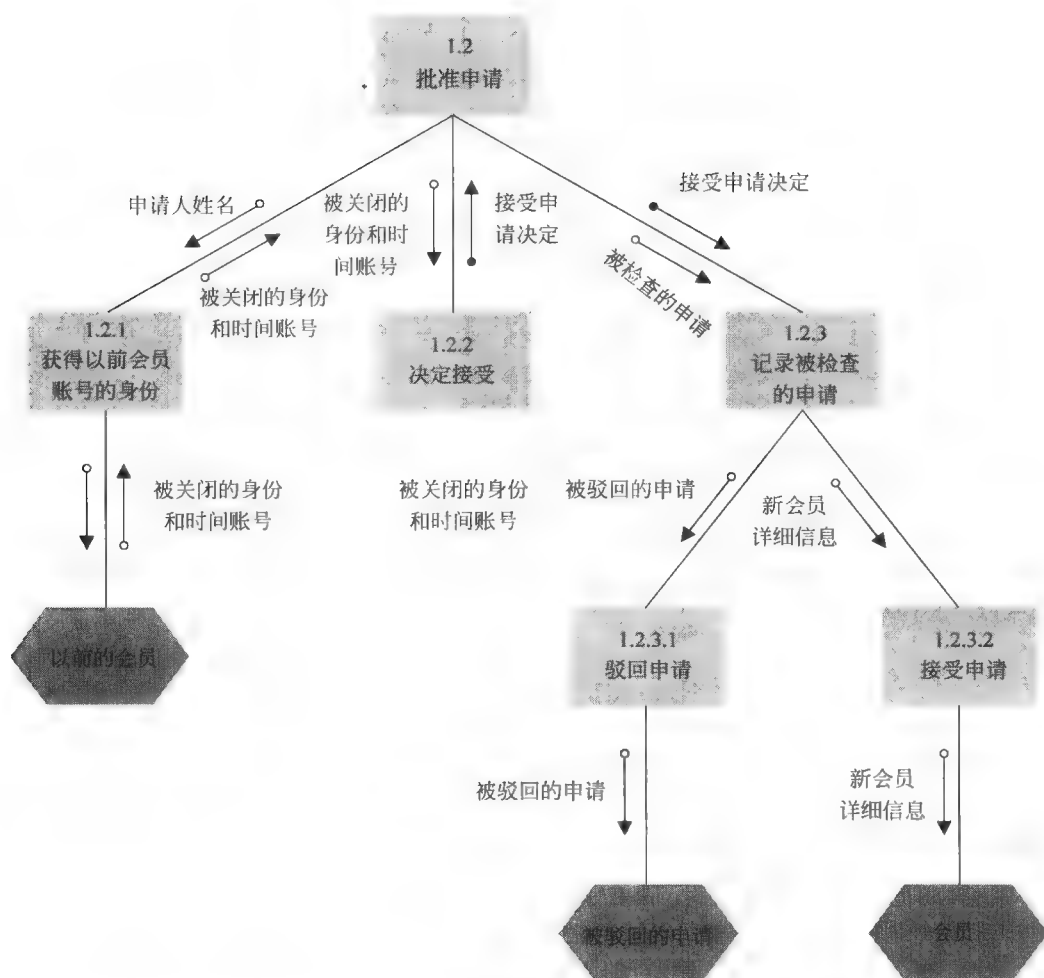


图 11-1 结构化设计的最终产品

- 原型化鼓励并要求最终用户主动参与，这增加了最终用户对项目的信心和支持。因为系统对他们来说显得很实际，所以用户的信心得到提高。
- 反复和修改是系统开发的自然结果。也就是说，最终用户总是想改变他们的想法，原型更好地适应了这种自然情况，因为它假定原型通过反复进化成为所需的系统。
- 据说，最终用户在看到需求被实现之前并不完全了解他们的需求。如果真是这样，原型化技术支持了这个原理。
- 原型是主动的（而不是被动的）模型，最终用户可以看到它、触摸它、感觉它和体验它。
- 获得认可的原型是书面的设计说明的运行等价，但有一个例外——错误能够很早地被发现。
- 原型化技术可以增加创造性，因为它促进了更快速的用户反馈，这可能会产生更好的方案。
- 原型化技术加速了生命周期的几个阶段。实际上，原型化合并了原先一个接一个的开发阶段。

原型化方法也有缺点，绝大多数缺点可以用一句话概括：原型化鼓励了通过生命周期的不良捷径。幸运的是，以下缺陷都可以通过合理地制定规则加以避免：

- 原型化鼓励了一种向过去曾经主导信息系统开发的“编码、实现和修改”生命周期的回归。正如许多公司已经了解到的，用原型化语言开发的系统可能存在与用 COBOL 这类语言开发的遗留系统同样的维护问题。
- 原型化没有否定对系统分析阶段的需要。原型与传统方式开发的系统一样能够轻易地解决错误的问题和机会。

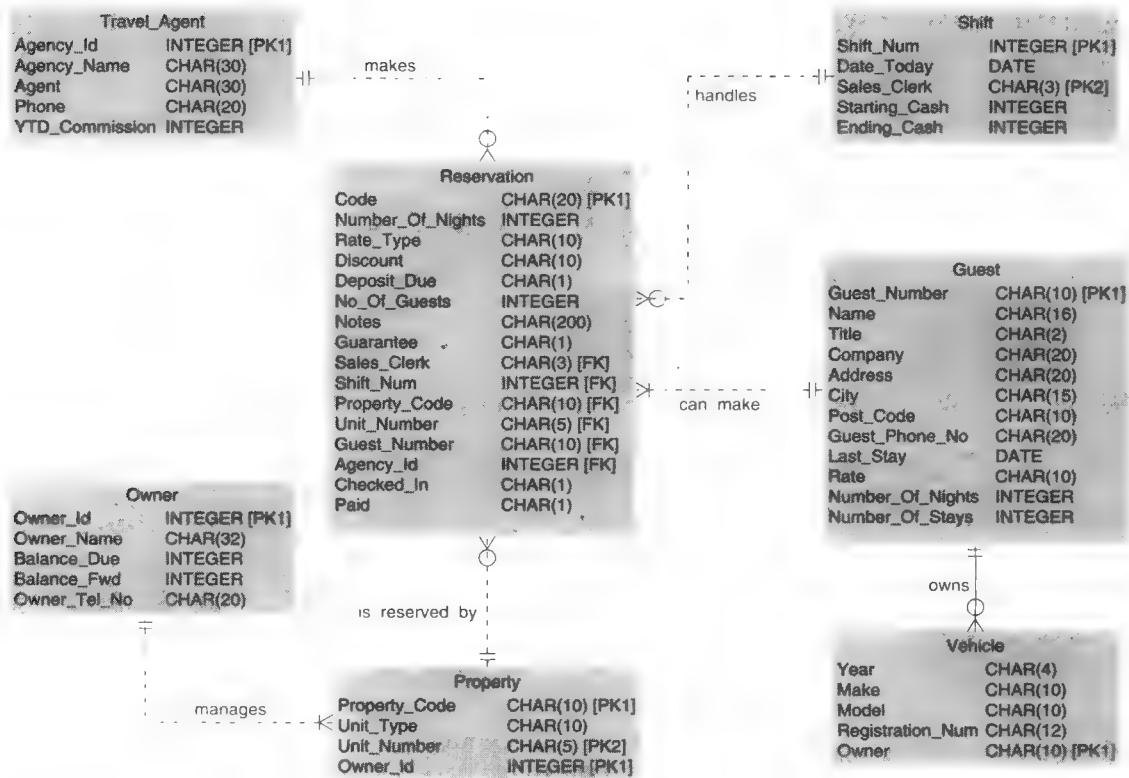


图 11-2 信息工程物理实体关系图示例

- 原型不可能完全替代书面说明。没有哪个工程师能够不用书面设计而完成一个引擎的原型，然而仍然有许多信息系统专家试图脱离说明来建立原型。原型化应该被用作其他方法学的补充（而不是代替）。可以降低书面设计的详细程度，但不能完全取消书面设计。
- 大量的设计问题没有被原型化所涉及。如果不小心的话，这些问题可能会被无意中忘掉。
- 原型化经常导致过早地提交一个设计（通常是开发出来的第一个设计）。
- 当建立原型时，系统的范围和复杂度可能会很快超出原先计划，这很容易失控。
- 原型化也会减少设计中的创造性。任何实现的基本特征——例如，报告的原型——都可能会阻碍分析员、设计人员和最终用户寻求更好的方案。
- 原型的执行经常比它们的第三代语言对应物慢得多（虽然这种差别正快速地消失）。

原型可以使用许多目前可以得到的 4GL 和面向对象编程语言快速地开发。图 11-3 显示了一个系统的原型屏幕。我们可以为简单的输出、计算机对话框、关键功能、整个子系统甚至整个系统建立原型。每个原型系统都由最终用户和管理人员检查，他们推荐需求、方法和格式，然后原型被更正、改进或精炼以反映新需求。原型化技术以一种相对直接的方式进行这种修订工作，修订和检查工作将一直继续下去直到原型被接受为止。最终，用户同时得到需求和实现那些需求的设计。

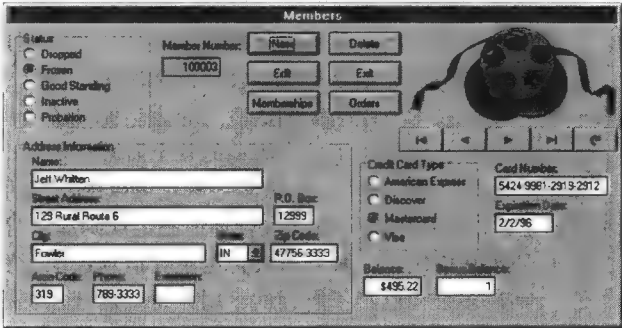


图 11-3 原型屏幕示例

11.2.1.4 面向对象设计

面向对象设计（OOD）是最新的设计策略，这个策略（和技术）背后的概念将在第17章中深入讲解，这里只进行简要介绍。这个技术是对第9章详细介绍的面向对象分析策略的扩展。图11-4为面向对象设计的许多模型图中的一幅。

对象技术试图消除“数据”和“过程”内容的分离。OOD技术用于精炼早期分析过程中确定的对象需求定义，并定义与设计相关的对象。

11.2.2 快速应用开发

另一个目前应用较多的设计策略是快速应用开发。快速应用开发（RAD）是各种结构化技术（特别是“数据”驱动的信息工程）与原型化技术和联合应用开发技术的结合，用以加速系统开发。

RAD要求反复地使用结构化技术和原型化技术来定义用户的需求并设计最终系统。通过使用结构化技术，开发人员首先构建业务需求的初始数据模型和过程模型；然后原型帮助分析员和用户验证那些需求，并正式地提炼数据和过程模型。模型、原型、模型、原型……这样的循环最终得出一个组合的业务需求和技术设计陈述，它们可以用于构造新的系统。

设计力量通过用户参与联合应用开发会议而得到增强。联合应用开发（JAD）（在第4章介绍并在第5章详细地讨论）是一种补充其他系统分析和设计技术的技术，它强调“系统所有者”、“用户”、“设计人员”和“构造人员”共同参与开发。在系统设计的JAD会议中，系统设计人员将扮演主持人的角色，会议可能是为期几天的研讨会，目的是讨论各种设计问题和交付成果。RAD之所以得到越来越多的重视，JAD作为一个主要因素，起到重要的作用。

11.2.3 系统设计策略

像大多数商业方法学一样，我们假设的方法学对系统设计方法不进行任何限制。相反，它综合了前面介绍的所有常用方法。音阶公司案例研究将在一个系统分析员的第一次任务的上下文中演示这些方法。这些系统分析技术将在如下的框架中应用：

- 你的信息系统构件（来自第1章）。
- 系统开发阶段（来自第2章）。
- 实现一个阶段的任务（在本章描述）。

给定这个上下文，现在可以研究系统设计了。首先将研究一个内部开发或者“构造”项目的系统设计；然后，将介绍当决策要求获取或者“购买”商业软件包时，系统设计阶段将受到什么样的影响。

11.3 系统设计之内部开发——“构造”方案

首先将一个内部开发项目的系统设计放到相对于系统生命周期的上下文中考虑。如图11-5所示，来自决策分析阶段的批准了的系统方案建议触发了这个设计阶段。设计阶段的目标有两个：第一，分析员寻求设计一个既满足需求又对用户友好的系统，人机工程学将在设计过程中扮演重要角色；第二，分析员向计算机程序员和技术人员提供清晰完整的设计说明。图11-5中被批准的设计说明将触发内部开发项目的构造阶段。

图11-6是一张任务图，描述了设计阶段应该完成的工作（=任务）。该图并没有强制采用任何特定的方法学，但我们将在相应的段落中描述那些可用于每个设计任务的方法、工具和技术。这张任务图仅仅是一个模板，项目团队和项目经理可以扩展或者修改这一模板以反映项目的特殊需求。

下面详细地介绍每个系统设计任务。

11.3.1 任务5.1——设计应用架构

第一个设计任务的目的是说明应用架构。应用架构按照数据、过程、接口和网络组件定义了一个、多个或者所有信息系统使用（和用于构造信息系统）的技术。因此，设计应用架构需要考虑网络技术，以及对系统的“数据”、“过程”和“接口”构件在业务地点之间的分布方式做出决策。

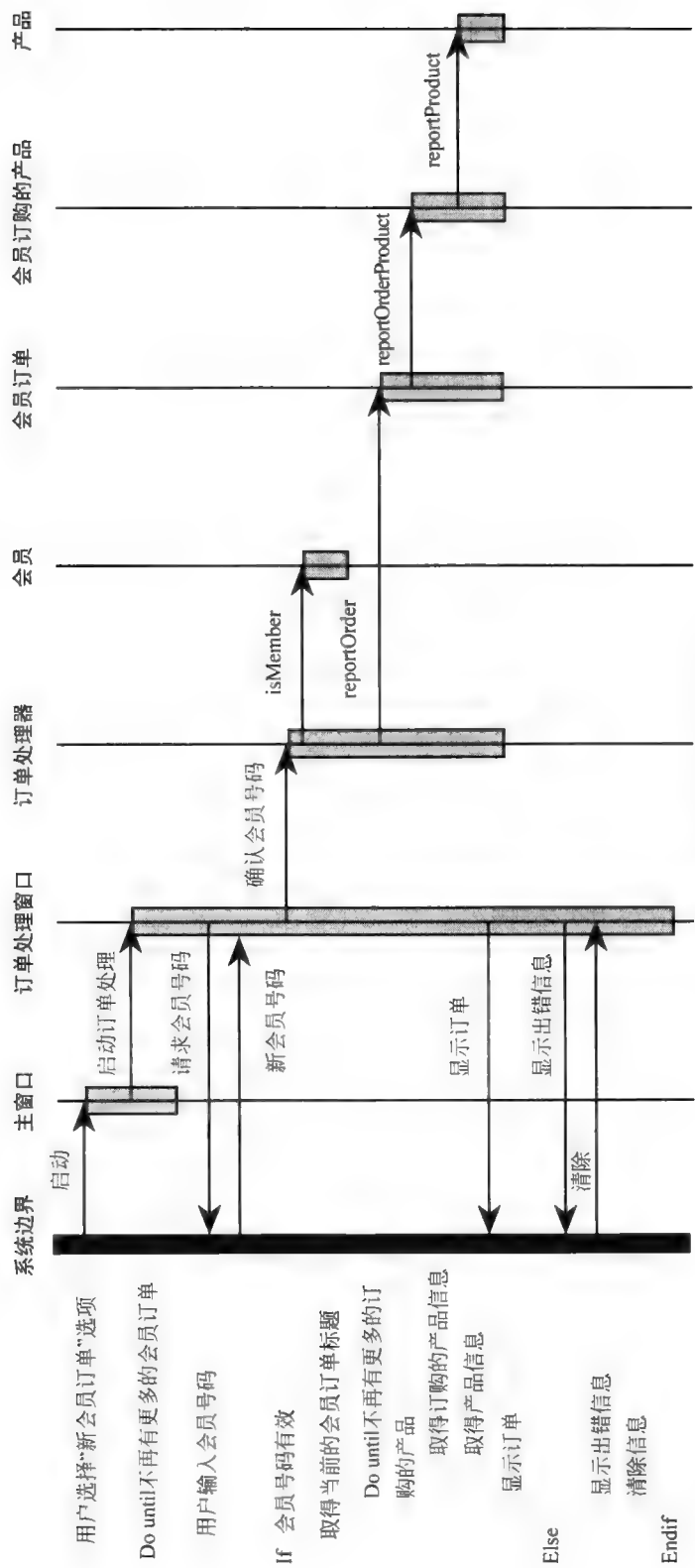


图11-4 面向对象设计模型示例

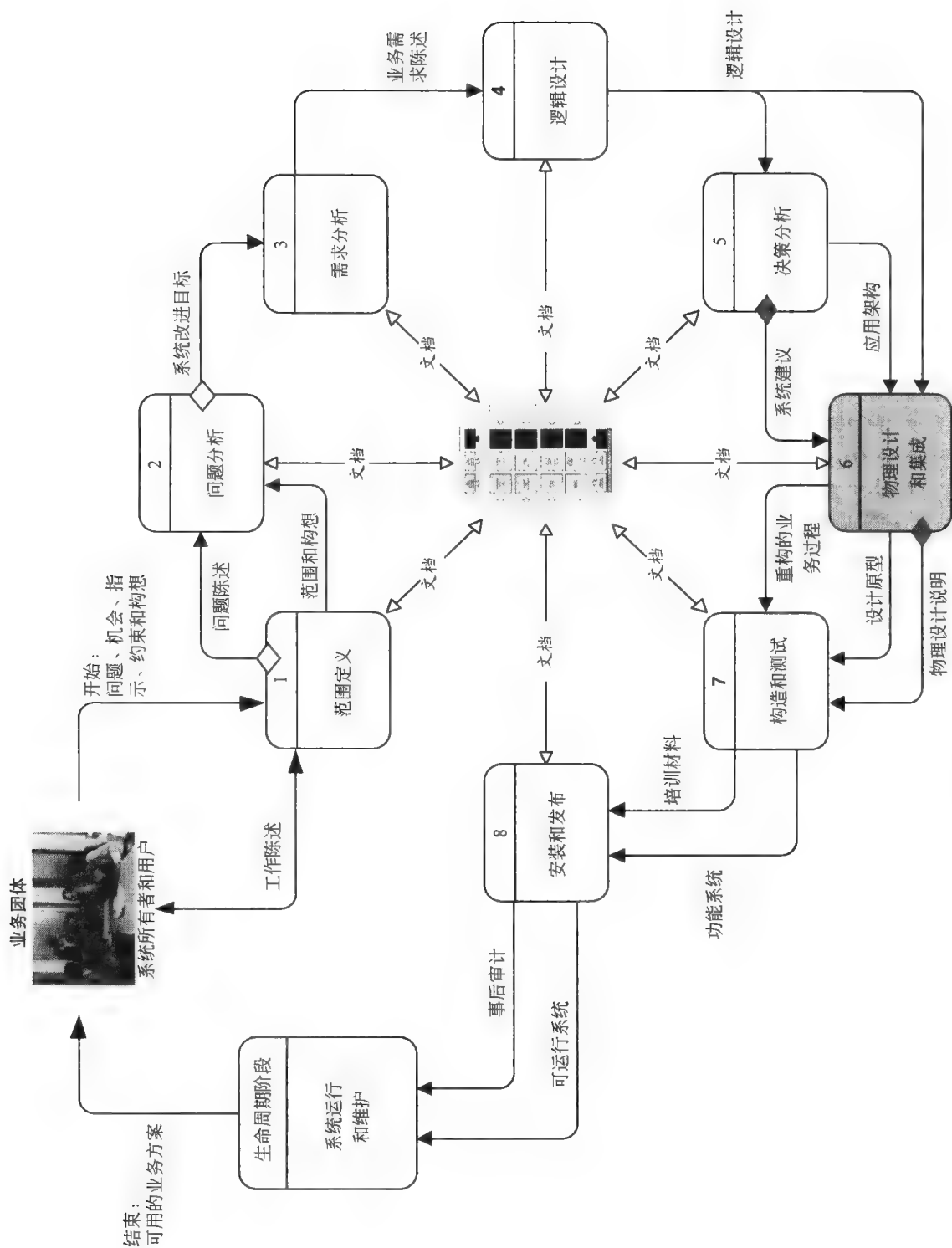


图 11-5 用于内部开发的系统设计上下文

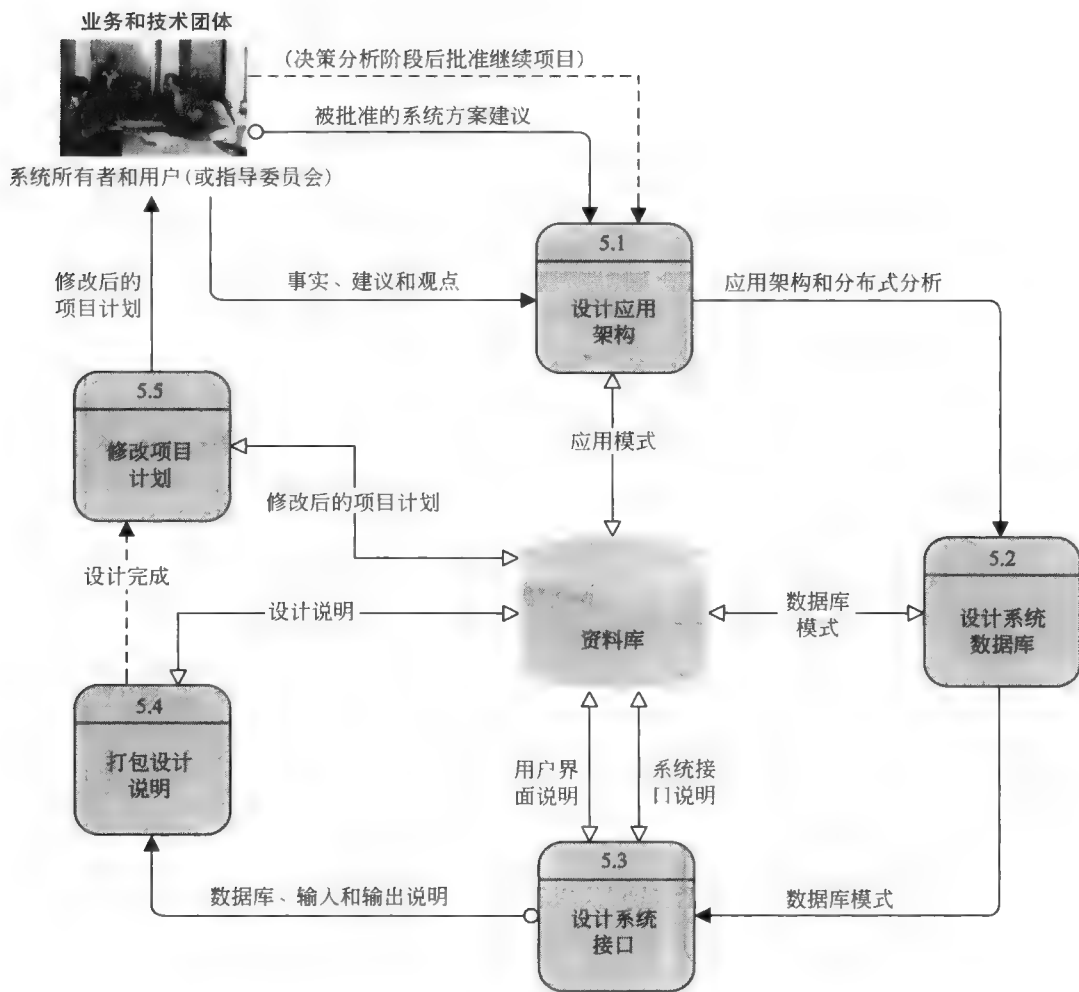


图 11-6 用于内部开发的系统设计任务

这个任务通过分析原先在需求分析期间创建的数据模型和过程模型实现。需要根据数据模型、过程模型和目标方案做出分布决策。当决定数据、过程和接口如何实现时，所有设计决策都要以文档形式记录下来。一种记录方法是使用物理数据流图（PDFD），它可用于确定网络之间的物理过程和数据存储（数据库）（见图 11-7）。将在第 12 章学习使用 PDFD 记录应用架构。

为了完成这个活动，分析员可以让一些“系统设计人员”和“系统用户”参与进来。系统用户可以参与到这个活动中，帮助说明业务数据、过程和地点问题。几个不同的“系统设计人员”专家可能对完成这个活动有帮助，包括数据和数据库管理员、网络管理员和工程师、应用管理员和其他专家（例如自动数据收集方面的专家可解决条形码技术和问题）。

这个任务的关键输入是各种来源的事实、建议和观点，以及决策分析阶段批准的系统方案建议。这个任务的主要交付成果是应用架构和分布式分析，它们可作为后续详细设计活动的蓝图。

11.3.2 任务 5.2——设计系统数据库

通常，下一个系统设计任务是开发相应的数据库设计说明。数据的设计绝不仅仅是对记录的简单布局。数据库是一个共享资源，一般会有许多程序使用它们，而且未来的程序可能会以一种以往无法想象的方式使用数据库。所以，设计人员在设计数据库时要使它适应未来的需求和扩展。

为了提高数据库性能，设计人员还必须分析程序将如何访问数据。你可能已经对各种程序设计数据结构及其对性能和灵活性的影响有所了解，这些问题影响着数据库组织结构的决策。数据库设计过

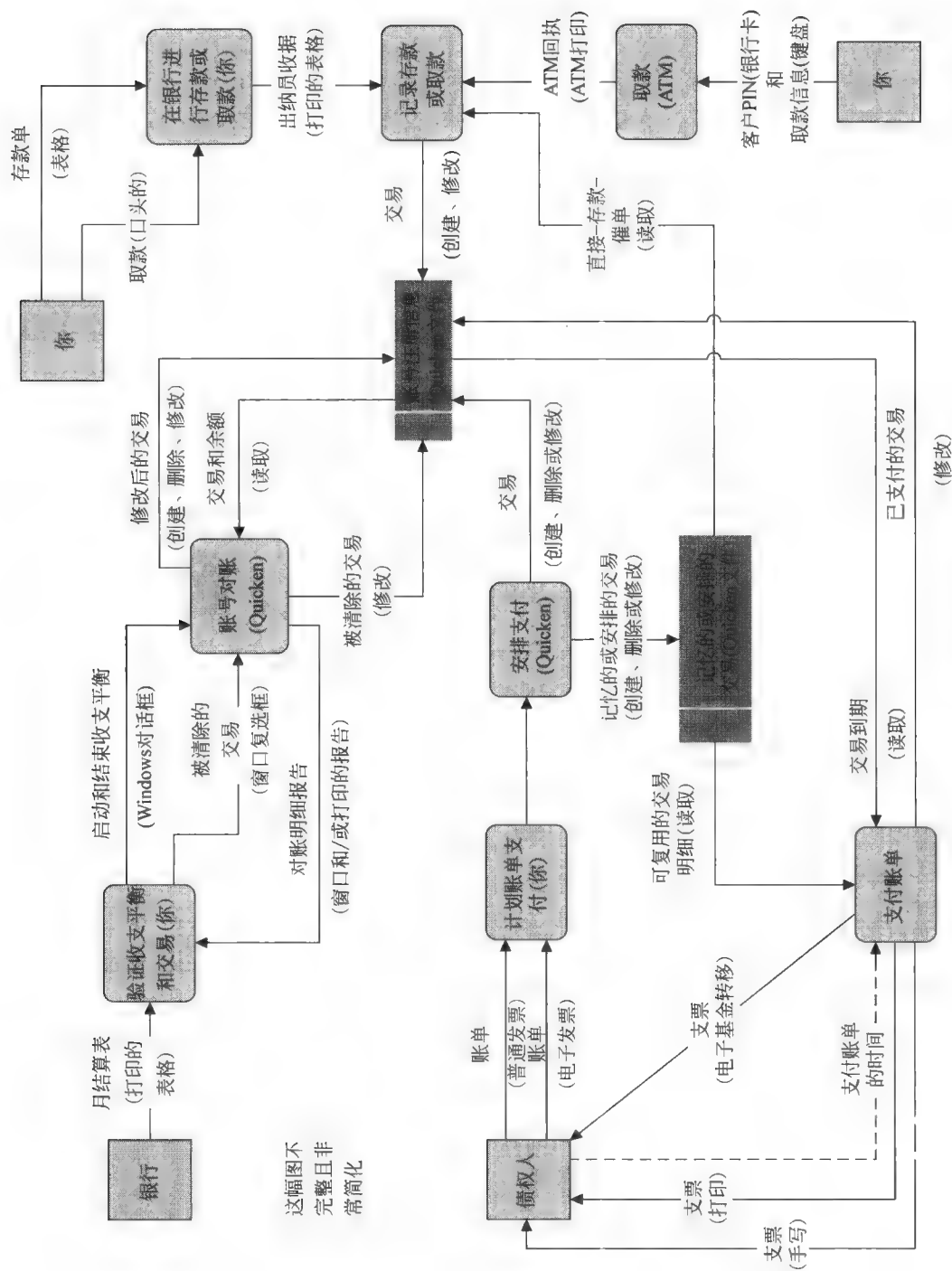


图 11-7 物理数据流程图示例

程中涉及的其他问题包括记录大小和存储容量需求。最后,因为数据库是共享资源,所以设计人员还必须设计内部控制,以确保在数据丢失或者损坏的情况下有必要的安全性和灾难恢复技术。

这个任务的目的是为能够适应未来需求和扩展的数据库准备技术设计说明。虽然可能由参与数据库建模的“系统分析员”主持这个任务,但负责完成这个活动的是“系统设计人员”。数据管理员可以参与(或者完成)数据库设计。新系统很可能会使用一个现有数据库的某些部分,在这时数据库管理员的知识尤为重要。最后,当需要为这个项目构造一个原型数据库时,“系统构造人员”也会参与。

如图 11-6 所示,这个活动的一个关键输入是来自前一个设计任务的应用架构和分布式分析决策,这个任务的交付成果包括得到的数据库模式。前面的图 11-2 显示了一个数据库模式的例子。数据库模式是数据库的结构模型,它描述了由数据库实现的记录和关系,将在第 13 章学习如何开发数据库模式。

11.3.3 任务 5.3——设计系统接口

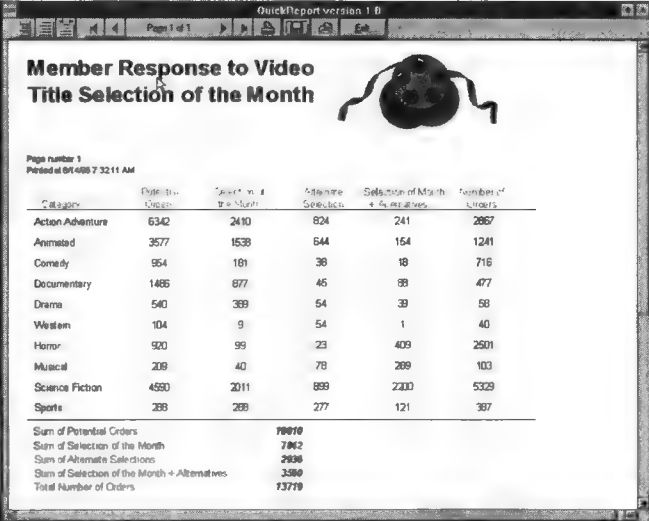
一旦设计了数据库,而且可能还构造了一个原型,系统设计人员就可以与系统用户一起开发输入、输出和对话的设计说明了。因为最终用户和管理人员必须同输入和输出打交道,所以设计人员必须仔细地征求他们的想法和建议,特别是关于格式方面的建议,而且还必须征求他们对于新系统易用性方面的想法和观点。

事务输出经常被设计为预打印的表格,其中事务明细将被打印在上面。报告和其他输出通常直接打印在纸上或者显示在一个终端屏幕上,但必须说明输出的精确格式和布局。最后,必须说明确保输出不丢失、不错误路由、不被误用或者保持完整性的内部控制。图 11-8 是输出设计的例子,将在第 14 章学习如何设计输出。

对于输入,设计系统使用的数据收集方法是至关重要的。例如,可以设计一个表格,在其中对输入的数据进行初始记录。你想简化数据在表格上的记录,而且也想简化数据从表格到计算机或计算机可读介质中的录入,如果数据由不熟悉业务应用的人输入,这一点就特别重要。而且,任何时候输入数据到系统中时,都可能会犯错误,所以需要定义编辑控制,以确保输入数据的正确性。输入原型屏幕的例子在图 11-3 中显示。将在第 15 章学习如何设计输入。

对于界面或对话设计,设计必须考虑这样一些因素,例如终端的熟悉程度、最终用户可能遇到的错误和误解、在某些地方对额外指示或帮助的需要以及屏幕内容和布局。需要预测最终用户可能犯的每个很小的错误或者可能按下的键——无论可能性多么小。此外,需要让最终用户容易理解屏幕上在任何给定时间显示的内容。图 11-9 是界面设计的例子。将在第 16 章学习如何进行界面设计。

“系统用户”应该参与设计这个活动。输入、输出和对话框是他们将要看到和使用的东西。包括原型化在内的设计工作都强调用户的参与,需要用户提供就每个输入/输出原型的反馈。“系统设计人员”负责完成这个活动,他们可以利用系统设计人



Category	Titles	Titles in Stock	Alternate Selections	Selection of Month + Alternates	Number of Orders
Action/Adventure	6342	2410	824	241	2667
Animated	3577	1538	644	154	1241
Comedy	954	181	38	18	716
Documentary	1485	877	45	88	477
Drama	540	389	54	39	58
Western	104	9	54	1	40
Horror	920	99	23	409	2501
Musical	209	40	78	269	103
Science Fiction	4590	2011	889	2300	5329
Sports	288	288	277	121	387
Sum of Potential Orders				78870	
Sum of Selection of the Month				7862	
Sum of Alternate Selections				2836	
Sum of Selection of the Month + Alternates				3360	
Total Number of Orders				13779	

图 11-8 输出原型屏幕示例

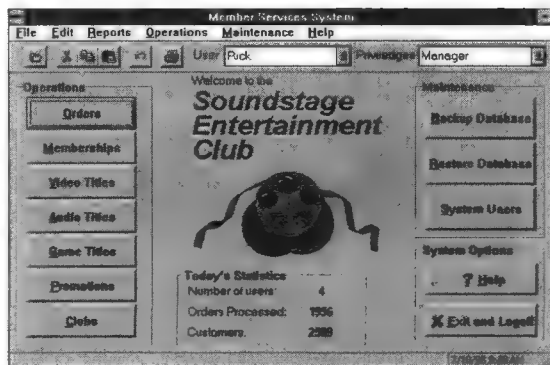


图 11-9 对话框界面原型屏幕示例

员在图形用户界面设计方面的专业知识。而且,“系统构造人员”在设计过程中可以通过原型化技术为用户构造各种屏幕设计以供检查。

如图 11-6 所示,这个活动的关键输入是来自前面任务的数据库模式,以及可以从项目资料库中得到的用户界面和系统接口说明。这个设计任务的交付成果是完成的数据库、输入和输出设计说明。

11.3.4 任务 5.4——打包设计说明

这个最后的设计任务包括把来自前面任务的所有说明打包成一套设计说明,并作为在系统开发方法学的构造阶段指导计算机程序员活动的设计说明。

但是,这个任务不仅仅是打包,具体还有多少工作取决于以下两点:1) 系统设计人员和计算机程序员的责任分界线划在哪里;2) 方法学和方案是否要求设计整个程序结构。大多数组织已经采用了加速系统开发方法,该方法不要求设计整个程序结构。程序结构需要处理质量问题,这些问题对那些使用老式程序设计语言的系统和通常基于大型主机的应用系统的开发人员有意义。

通常由“系统分析员”(可能有“系统设计人员”辅助)完成这个任务。在进行设计说明封装和构造阶段之前,系统设计应该被所有相应人员检查。虽然“系统用户”已经看到并认可了新系统的输出、输入和对话框,但仍需对新系统的工作流和数据流进行最后一次检查和认可。“系统所有者”应该保留最后的机会质疑项目的可行性,并决定项目是否应该被调整、终止或者批准继续构造。在项目的这个阶段,公司的审计人员将深入参与,他们将对新系统中的内部控制做出判断。

如图 11-6 所示,这个任务的输入是前面创建的各种数据库、输入和输出设计说明。一旦这些设计说明得到检查、认可并组织成适合于构造新系统的设计说明,就可通过项目资料库将它们提供给系统构造人员。常见的情况是项目经理通过共享资料库提供设计说明,而不是给每人一套组织好的设计说明打印件。

11.3.5 任务 5.5——修改项目计划

现在已接近完成设计阶段,我们还应该重新评估项目的可行性,并相应地修改项目计划。项目经理和“系统所有者”以及整个项目团队一起主持这个任务,“系统分析员”和“系统所有者”是这个任务中的关键人物。根据已完成的设计工作,分析员和所有者应该考虑整个项目的进度、费用估计以及其他可能需要调整的估计。

如图 11-6 所示,当项目经理认为设计完成了时,这个任务就被触发。这个任务的关键交付成果是修改后的项目计划,它现在应该包括了构造阶段的详细计划。修改项目计划的技术和步骤已经在第 4 章中讲授过。

11.4 系统设计之集成商用软件——“购买”方案

本节介绍方案中包含获取商用现成产品(COTS)软件产品的系统设计。图 11-10 说明了包含采购或“购买”方案的项目的生命周期。注意业务需求陈述(对于软件)和业务集成方案触发了一些在我们刚刚学习的内部开发过程中没有的阶段。“购买”和内部开发项目之间最显著的区别是包含了一个采购阶段和一个涉及软件和服务的决策分析阶段(标记为 5A 的过程)。

当需要新软件时,选择合适的产品经常很困难,决策由于技术、经济和政策考虑而变得复杂。差的决策可以毁掉其他方面都很成功的分析和设计。系统分析员越来越多地参与软件包(以及支持分析员正开发的特定应用系统的外设和计算机)的采购。

在本节中,将介绍完成一个“购买”方案的采购和决策分析阶段需要涉及的任务。如图 11-10 所示,“购买”方案影响了生命周期的其他阶段的完成方式(受影响的阶段用浅灰色阴影表示)。介绍了采购和决策分析阶段之后,将探索“购买”方案对其他阶段的影响。

图 11-11 是一个任务图,它描述了为完成一个“购买”方案的采购和决策分析阶段应该进行的工作(=任务)。该图并没有对专门的方法学进行任何限定,但我们将相应段落中描述那些可用于每个设计任务的特定方法、工具和技术。这个任务图仅仅是一个模板,项目团队和项目经理可以扩展或者修改这个模板,以反映项目的特殊需求。

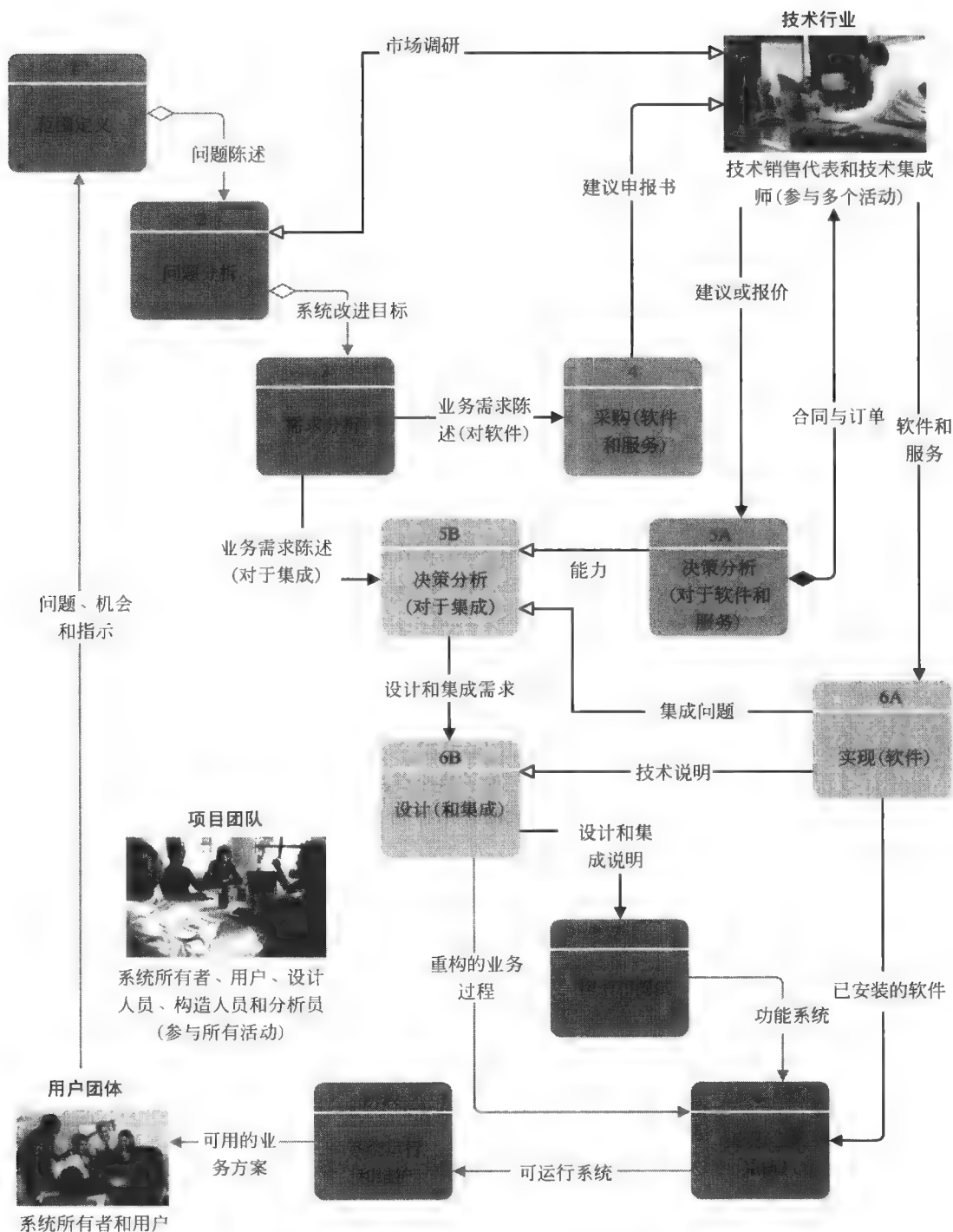


图 11-10 商用现成产品软件方案的系统设计上下文

研究工作还应该确定能够提供所需产品的潜在供应商。分析员完成了分析工作以后，将开始同这些供应商谈判。因此，在进行了研究之后，分析员能够更好地对付供应商的销售策略！

这个任务的目的是研究技术替代方案，对重要的准则加以说明，这些准则对将被选择的新硬件和/或软件很重要。由项目经理主持这个任务，“系统设计人员”负责完成这个任务，设计人员可以从不同技术专家那里寻求帮助，包括数据管理员和数据库管理员、网络管理员和应用管理员。

如图 11-11 所示，这个任务的关键输入是在需求分析阶段建立的业务需求陈述（对于软件）。设计人员还将从各种来源获得其他产品和供应商信息，注意不要仅仅从一个销售人员那里获得信息——并不是这个销售代表不诚实，而是销售人员的首要原则就是强调产品的优点而不强调它的弱点。这个任务的主要交付成果是一个潜在供应商、产品选择和技术评价准则的清单。

为了完成这个任务，设计人员必须进行深入的研究，以获得关于硬件/软件产品和供应商的重要信息。还必须仔细地检查各种信息来源，确定可获得产品的潜在供应商。如果公司已经有了一个承诺或者合同，规定从某个特定的来源获得某种产品，这一步就可以省略。最后，设计人员必须检查产品、供应商和供应商调查结果。

11.4.2 任务 4.2——向供应商征求建议（或报价）

下一个任务是向供应商征求报价建议。如果公司只从单一货源（例如 IBM）购买，那么这个任务就是十分随意的，只需直接联系供应商并索取报价单和服务条款即可。但大多数决策有许多货源需要选择，这时，最好通过市场竞争实现最大利益。

这个任务要求准备以下两个文档之一：**报价申报书（RFQ）**或者**建议申报书（RFP）**。当已经决定使用某个特定的产品，但那个产品可以从几个分销商处获得时，就应该使用报价申报书，它的主要目的是征求特殊的配置、价格、维护承诺、购买者提出的条件和服务。当有几个不同的供应商和/或产品备选，而你想征求竞争的建议方案和报价时，就应该使用建议申报书。RFP 可以看作是 RFQ 的超集，它们都定义了用于验证的选择准则。

RFP 的主要目的是同潜在的供应商沟通需求和期望的特征。需求和期望的特征必须分为强制性的（供应商必须提供的）、特别重要的（希望供应商提供的，但也可以内部获得或者从第三方供应商处获得的）或者期望的（没有也可以工作的）。需求还可以按照另外两个标准分类：满足系统需要的需求和满足对供应商的需要的需求（例如服务）。

这个任务由项目经理主持，由“系统设计人员”负责完成，当编写 RFP 或 RFQ 时，还可以从数据管理员和数据库管理员、网络管理员和应用管理员那里寻求帮助。

这个任务的关键输入是从前面的研究得出的潜在供应商、产品选择和技术评价准则，主要交付成果是候选供应商收到的 RFP 或 RFQ。RFP 的质量对最终建议的质量和完整性有重要影响。图 11-12 为一个 RFP 的提纲，实际的 RFP 太长，这里不详细列举。

在第二部分中获得的许多技能（例如过程建模和数据建模）可能对在 RFP 中沟通需求很有用。供应商十分善于利用这些工具，因为他们发现这样更容易匹配产品并得出满足需求的建议。其他重要的技能包括撰写报告（在第 10 章讨论）和调查表（在第 5 章介绍）。

11.4.3 任务 5A.1——验证供应商的声明和性能

把 RFP 或 RFQ 发送给潜在供应商后不久，你就会收到建议和/或报价。对建议不能也不应该只看其表面价值，必须验证其声明和性能。这个任务针对每个建议独立操作，建议之间不进行比较。

这个任务的目的是验证所收到的供应商建议和/或报价。由“系统设计人员”负责完成，设计人员让以下人员参与验证建议：数据管理员和数据库管理员、网络管理员和应用管理员。

这个任务由接收来自潜在供应商的建议和/或报价单触发，关键输出是建议或声明经证明是有效的，那些供应商建议，同时声明无效的建议。

为了完成这个任务，设计人员必须收集并检查关于产品需求和特征的所有信息。他们必须检查供应商的建议，并排除不能满足所有强制性需求的建议。当然，如果清晰地陈述了需求，应该不会有供应商提交这样一份建议。对于不能满足一个或多个特别重要需求的建议，可以通过其他方式实现那个

需求或特征的验证。对于没有被排除的每个建议，设计人员必须根据验证准则验证供应商的声明和承诺。关于强制性需求、特别重要需求和期望需求的声明和特征可以通过供应商填写的调查表和检查表（包括在 RFP 中）验证，并使用供应商提供的用户手册和技术手册作为参考；承诺则只能通过确保它们写在合同中进行验证。最后，性能最好通过一个演示来验证，这对评价软件包尤其重要。演示使你能获得用来证实能力、特征和易用性的测试结果和调查结果。

建议申报书（RFP）	
I. 引言	III. 需求和特征
A. 背景	A. 硬件
B. 需求概述	1. 强制性需求、特征和准则
C. RFP 文档解释	2. 基本需求、特征和准则
D. 要求供应商的行动	3. 期望的需求、特征和准则
II. 标准和指示	B. 软件
A. 导致合同的事件进度	1. 强制性需求、特征和准则
B. 选择决策的基本原则	2. 基本需求、特征和准则
1. 谁应该和谁谈话，什么时候谈	3. 期望的需求、特征和准则
2. 谁支付什么	C. 服务
3. 建议要求的格式	1. 强制性需求、特征和准则
4. 对演示的期望	2. 基本需求、特征和准则
5. 对合同的期望	3. 期望的需求、特征和准则
6. 期望的参考	IV. 技术调查表
7. 对文档的期望	V. 结论

图 11-12 建议申报书

11.4.4 任务 5A.2——评价和分级供应商建议

现在可以对有效的建议进行评价和分级了。实际上，评价和分级是另一种在系统开发期间所进行的成本效益分析。在进行实际的评价之前，应该建立评价准则和打分系统，以避免由于下意识地关照某个建议而使准则和打分不够公正。

理想情况下，这个任务应该由主要负责人主持，由“系统设计人员”负责完成，设计人员可以邀请一些专业人员参与评价和分级建议，包括：数据管理员和数据库管理员、网络管理员和应用管理员。

这个任务的输入包括有效的建议和用来分级建议的评价准则，关键交付成果是硬件和/或软件建议。

进行可行性评估的能力对于完成这个任务来说特别重要。可行性评估技术和技能在第 10 章已经进行了介绍。为了完成这个任务，设计人员必须首先收集和检查与有效的建议相关的所有细节，然后建立一个评价准则和打分系统。有许多方法可以实现这一点，有些方法建议以点数权衡需求，更好的方法则建议使用美元！对管理层来说，货币系统比点数更有说服力。这种技术是以硬通货和软通货为基础评价建议，硬通货成本是必须支付给所选供应商的设备或软件的费用，软通货成本是如果选择一个特定的供应商，将支付的额外费用（例如，如果选择供应商 A，可能需要向供应商 B 支付额外的费用，以弥补供应商 A 的建议系统中的不足）。这个方法将会促成与能够实现所有基本需求而同时又可提供最低总硬通货成本及软通货成本（经过时间调整的）的供应商签订合同（参看本章推荐读物的 Isshiki 1982 或者 Joslin 1977，其中对这个方法的详细解释）。一旦建立了评价准则和打分系统，完成任务的最后一步就是对供应商的建议进行实际评价和分级。

11.4.5 任务 5A.3——签订合同并听取供应商汇报

如果已经对供应商建议进行了分级，下一步活动通常包括向管理层提供一个建议，供最后认可。如果分析员要说服管理层采用建议的话，沟通技能（特别是推销技能）就很重要。根据管理层对建议的批准，会与获胜的供应商签订一个合同。这个活动经常还包括听取被淘汰供应商的汇报，以避免失

去回转余地。

这个活动的目的是与提供获胜建议的供应商通过谈判签订合同，并听取那些提交了被淘汰建议的供应商的汇报。理想情况下，主要负责人（负责批准建议和项目继续进行）应该主持这个任务，但是“系统设计人员”必须做出建议，并捍卫其建议及确定合同。在做这些工作时，系统设计人员可以让公司律师来起草合同。撰写报告和汇报技能对完成这个任务很重要。

关键输入包括硬件和软件建议以及来自前一个任务的没有得到确认的建议。在等待主要负责人的批准期间，合同订单将最终提供给“获胜的”供应商，建议的汇报则将提供给被淘汰的供应商。

为了完成这个任务，设计人员必须首先提供一个硬件和软件建议供最后批准。一旦做出了最后的硬件和软件批准决策，就必须与获胜的供应商协商合同，某些特殊的条件和条款可能必须写入标准合同和订单中。理论上，没有律师的建议，不应该签署计算机合同。分析员必须仔细地阅读并澄清所有的版权合同。没有一个有资格的会计师和管理人员的同意，不应该批准任何最终决策，因为购买、租借以及带购买的租借方式涉及复杂的税收问题。最后，为了保持公共礼貌并维护良好关系，给被淘汰的供应商提供一次建议汇报。这次会议的目的不是给供应商第二次机会获得合同；相反，听取汇报的目的是告知被淘汰的供应商其建议和/或产品中的确切弱点。

11.4.6 购买决定对剩余生命周期阶段的影响

仅仅购买或构造实现了系统需求的系统还是不够的，分析员必须将新系统集成或接口到其他各种对企业重要的现有系统中，这些系统很可能会使用差别很大的技术、技巧和文件结构。

分析员必须考虑目标系统如何适应目标系统联盟，集成需求对确保目标系统与这些系统的兼容性是至关重要的。

如图 11-10 所示，购买一个商业软件包方案的决定可能会影响生命周期的其他阶段（用浅灰色阴影表示）。通过决策分析（对于软件和服务）阶段对商业产品进行评价，我们了解了产品的性能（或缺点）。在集成的决策分析过程中，需要做出修订以反映包括在业务需求陈述的数据模型和过程模型中的新知识。当收到供应商的软件和服务时，软件必须被实现。在实现过程中，可能会遇到集成问题，这些集成问题也必须被反映到业务需求陈述中。这些性能和集成问题反映在设计 and 集成需求中。

最后，给定设计和集成需求后，现在必须完成设计阶段。设计阶段的任务与本章前面讨论的许多任务相同，主要的差别在于不是简单地“开发”整个系统。而是，可能要为一些组件设计技术说明，这些组件包括开发程序的一个小子集、软件工具以及为了业务过程和商业软件产品能正确地集成并一起工作所需的其他组件。下面考虑一个例子，现有的业务系统可能使用了条形码技术收集数据，但是软件产品可能需要通过键盘输入数据，这时就需要定制软件产品，以允许数据通过键盘输入或者从批处理文件获得扫描的数据。

复习题

1. 系统分析和系统设计的基本差别是什么？
2. 有哪些不同的模型驱动方法？
3. 原型化有什么优点？
4. 内部构造一个开发项目中进行系统设计包括哪 5 项高层任务？
5. 为什么需要设计应用架构？
6. 在设计系统数据库时，设计人员应该始终关注什么？
7. 什么是数据库模式？
8. 当设计系统界面时要达到的目标是什么？
9. 当设计系统界面时设计人员应该关注哪些特定因素？
10. 如果软件是购买的而不是内部开发的，在系统设计中需要哪些阶段？这些额外的阶段的目的是什么？
11. 什么是报价申报书（RFQ）？

问题和练习

1. 系统设计的主要目标是什么？系统设计中包括哪些阶段？如果系统分析完成得不好或者不完整，好的系统设计能够弥补吗？
2. 匹配第 1 列中的定义和第 2 列中的例子。

1. 信息工程	A. 信息工程
2. JAD	B. 结构化的设计模块原型
3. 现代结构化设计	C. 强调参与开发
4. 原型化	D. IBM 的 Rational
5. 系统设计	E. 结构化设计的导出模型
6. 物理实体关系	F. 组合的数据和过程
7. 耦合和内聚	G. 模型驱动的、数据为中心的、过程敏感的技术
8. RAD	H. 强调图形化的系统模型
9. 模型驱动设计	I. 使用 RAD 构建功能不完整的模型
10. 编码、实现和修改	J. 过程分解技术
11. 基于资料库的 CASE 工具	K. 基于计算机的方案说明任务
12. OOD	L. JAD、原型化和结构化技术的混合
13. 结构图	M. 潜在的原型化缺点

- 原型化有许多优点，但也有一些缺点和不足。讨论这些缺点和不足。可以通过运用什么策略来减少它们出现的风险？
- 考虑问题 3 提出的问题，撰写一份关于原型化的一到两页的策略和程序备忘录，分发给你所在企业中的所有的系统分析员和设计人员。
- 你是一个企业的系统设计人员。你的团队中的另一位设计人员最近退休了。你的经理找你谈话想让你代替他的工作。当你进行谈话时你应该如何表现？
- 你正在为一个开发中的新系统设计一个数据界面屏幕。这个数据界面屏幕的用途是输入司机提交给州政府 DMV 的地址变更信息。每个主数据操作员每天都会在这个屏幕上输入手写表格中的 1000 个地址变更信息。要记住的一个十分重要的原理是什么？
- 在你的企业中，当设计说明通过后，给项目涉及的每一个人一份打印的设计说明是一项传统。这花费很多，但管理层觉得这是感谢每个人在项目中的贡献并保持他们的投入的一种方式。如果企业不在意这些费用，这件事有什么不妥吗？

项目和研究

- 你是一名系统分析员，在一个橱柜制造公司的 IT 商店工作了好几年。公司由于其产品质量和领导地位而出名。前任首席信息官（CIO）毕业于

- 你是一位负责检查供应商的建议方案的系统设计人员。一个曾经给你们公司做过令人满意的工作的供应商提交了一份建议，但该建议没有满足几个关键的需求。你应该怎么办？
- 匹配第 1 列中的定义（或例子）和第 2 列中的术语。

A. 采购阶段	1. DBA
B. 显示通过网络的物理过程和数据库	2. 分布式分析
C. 被代替的基于文本的显示	3. 应用架构
D. 供应商评估标准和评分系统	4. 数据库模式
E. 竞争的建议文档	5. RFQ
F. 详细设计阶段活动的部分蓝图	6. 审计员
G. 负责数据库架构的专家	7. COTS
H. 对特殊产品的请求文档	8. PDFD
I. 负责内部控制的专家	9. 硬通货成本
J. 用于构造信息系统的技术	10. “购买”方案
K. 数据库的结构化模型	11. GUI
L. 商业软件产品	12. RFP

- 你正在开展的一个项目的生命周期包括一个“购买”方案，为公司的市场专家购买一个商业现货产品。你的公司想得到其他竞争建议。使用图 11-12 显示的格式准备一份建议申请书（REP）（注意：由于练习的目的，不需要开发一个完整详细的 RFP，但你的 RFP 至少应该包含每一节需要的高层细节和信息）。
- 你为一个咨询公司工作，该公司被要求做项目的系统设计部分。系统分析部分由另一个咨询公司完成。在系统设计过程中，你确定需求中有一个错误。你不确定错误有多严重，但你能确信：如果这个错误是由另一个公司造成的，而被项目经理指出来，系统设计工作将不得不停止，直到修正错误。这将会使你的公司落后于进度，而且当修正错误时，公司不会给你和其他咨询员付工资，也不会让你们休息。在这种情况下你的道德取向是什么？
- 数据安全和隐私是越来越重要的问题。系统分析员、系统设计人员和数据库管理员在开发和维护一个关系数据库系统时需要了解哪些安全和隐私事项。

“老式学校”，最近退休了，由一位新的有生气和进步的 CIO 接替。新的 CIO 为了提高企业的成熟度水平，正在进行一系列的讨论会以开发企业的

第一个 IT 架构规划。你和其他的系统分析员和 design 人员每个人都被要求就企业应该采用哪种系统设计方法作为标准方法这个问题提供意见。使用本书中的信息、你自己的工作经验和你进行的其他补充调研，给你的 CIO 写一份备忘录：

- a. 提供与你所在企业相关的背景信息——例如，企业远景、任务、战略目标、成熟度水平、组织结构和文化。
 - b. 描述不同的系统设计方法。
 - c. 比较这些方法的优缺点。
 - d. 推荐一个特定的方法或者一组方法作为你企业的标准方法。建议应该包括对你提出的建议的论证。
2. 你是一个企业级项目的大型系统分析员和 design 人员团队中的一员，该项目涉及你所在企业的每个部分，既包括公司总部机构，也包括分布在全国的地区机构。按照职员们的建议，执行管理层已经决定在内部进行系统设计。由于项目和你的企业的规模较大，这个项目需要涉及上百个系统所有者和用户的参与，他们分散在总部和地区机构中。对这些系统所有者和用户中的许多人来说，这是他们第一次参与到这种项目中。你的责任之一是确保他们理解其在这个阶段所扮演的角色，以及对整个项目成功的重要性。
- a. 给你所在企业的系统所有者和用户写一份电子邮件，内容是设计阶段和他们在该阶段扮演的角色，使用任务 5.1 ~ 5.5 作为指南。你的目的是确保他们理解自己的角色以及对项目成功的重要性。
 - b. 当你写完电子邮件后，向你所在企业中的几个人解释其内容。他们读完后认为你的邮件清楚、完整并具有说服力吗？结果如何？如果这是真实情况，他们会理解自己的角色吗？你的解释会对他们的投入产生正面影响吗？
3. 在第 1 题中，要求你寻找一些系统设计方法，包括原型化。最近几年中，原型化和应用工具的数量呈指数增长。在因特网上研究一些可得到的不

同的原型技术。另外，同几个使用原型化的系统设计人员交谈，询问他们对市场上的不同原型设计工具的看法。准备一份书面报告分析描述你的研究工作并报告你的发现。

4. 你在一个销售企业的 IT 商店工作，该企业在你所在州有好几个卫星机构。企业想开发一个基于 Web 的信息系统，以便其卫星机构可以实时提交他们的销售报告。但你的 IT 商店很小，每个人都已经完全地投入到维护和支持活动或者其他项目中，除此之外，没有一个人拥有开发基于 Web 的信息系统的经验。所以管理层决定外包设计和开发。你的工作是：
 - a. 同你本地社区的 IT 供应商交谈，了解他们在请求建议书（RFP）方面的经验；也就是说，寻找他们在 RFP 中遇到的常见缺陷，哪些关键事情需要被包含其中，以便准备一个合适的建议。给你的管理层准备一份简短的备忘录，描述你的会谈。
 - b. 研究一些可得到的不同的 RFP 模板，包括图 11-12 使用的模板。选择一个，并解释原因。
 - c. 使用你选择的模板，撰写一份请求建议书。
5. 现在你已经完成了上一题中的请求建议书，你的下一个任务是计划这个项目的系统设计阶段。使用你目前已经读过的关于系统设计的内容，准备一份高层项目计划，显示主要任务、资源和需要的估计小时数、时间帧，以及依赖关系（参考第 3 章）。
6. 大量的评估准则和评分系统被公共和私营企业用来评估和分级供应商的建议。例如，加利福尼亚州有一个可选的“最佳值”方法，州政府部门用它进行分级和选择供应商。另外，本书在建议读物清单中参考了有关其他方法的书籍。在因特网上研究公共和私营企业使用的这些方法。另外，试着同这些企业中负责评估和分级供应商建议的职员，以及准备这些建议的职员交谈。

小型案例

1. 在前面章节中，你为一个政府部门设计一个系统。挑选那个系统创建过程中的一个特定的任务（例如，开发一个网站），并通过至少从两个不同的供应商为该任务收集两个建议。验证供应商提供给你的说明和指标。分析你的发现，并将你的结果和建议提交给你的教授。
2. 你正在为一个大型公司开发一个复杂的系统。代

码将很复杂，编程语言对于编程团队来说相当新鲜。你的老板要求你在开发中使用一个进化原型。这是可用的合适的原型模型吗？如果不是，你应该如何处理老板的请求？

3. 在第 7 章，你对一个汽车租赁机构进行了研究，并为它创建了一个数据模型。利用你前面所做的工作，以及初步的交谈，为汽车租赁系统创建一

个原型。

注意（对学生）：你为什么需要进行更多的交谈？这是为了使你能够开发一个用户希望的界面以及功能吗？记住，除了过程功能，用户将决定系统的成败。他们想让界面是什么样子？他们想用什么格式处理输入的数据？

4. 在上一题目中，基于前一章的工作和初步的交谈，你为一个汽车租赁系统创建一个原型。现在你已经完成了原型的创建，回到你的客户那里，向他们展示你的原型。

团队和个人练习

1. 进行一个练习来提高创造力。可以是一个谜语、一个艺术项目、一次冒险等。把你的想法提交给教授。然后教授将把这个创造性练习拿到班上，以便每个人进行练习，但没有人进行他们自己提出的练习。完成你被赋予的任务。
2. 访问几个 Web 站点，记录你特别喜欢（和不喜欢）的特点和界面特征。把这些信息拿到班上分

- a. 客户有什么反应？客户喜欢这些功能吗？界面设计呢？记录他们的回答，以及肢体语言。
- b. 重做你的系统以满足客户的建议和希望。客户想要的东西中有不合理的吗，或者有目前不可行的吗？如果有，请记录下来。
- c. 提交你的初始原型和改进后的系统原型，并提交一篇短文，讨论客户需求以及他们对你的原型的反应，在改进系统中你对他们的建议涉及多少，以及其他关于你的系统的背景信息。

享。考虑到每个人的经验，你认为对一个网站来说什么是特别好的特点？它不应该有什么特点？

3. 假设你是一个团队的领导，这个团队总是落后进度。假设进度推迟是由于计划和时间管理的问题，而不是资源分配问题。不能要求你的经理重组团队，你可以做什么来鼓励团队成员满足进度？

应用架构和建模

本章概述和学习目标

本章讲授设计信息系统应用架构的技术，重点是物理过程模型。信息系统应用架构和物理过程建模包括各种将知识、过程和通信分布到一个分布式计算环境中的技术。物理数据流图使用设计单元的形式记录应用架构和设计——设计单元是特定地点的数据和过程的内聚集合——它可以进行更详细的设计、原型化或构造，最终被实现成一个独立的子系统。本章将介绍以下内容：

- 按照“知识”、“过程”和“通信”构件（所有信息系统的构件）定义一个信息系统架构。为了同现代趋势保持一致，将这些构件分布到“网络”上。
- 区分逻辑数据流图和物理数据流图，并解释如何使用物理数据流图建模信息系统架构。
- 描述信息系统设计的集中式和分布式计算方案，包括各种客户/服务器和基于因特网的计算方案。
- 为信息系统设计描述数据库和数据分布方案。
- 为信息系统设计描述用户界面和系统接口方案。
- 为信息系统设计描述各种软件开发环境。
- 描述用于开发或确定信息系统架构的策略。
- 为信息系统架构和过程绘制物理数据流图。

本章关键术语

应用架构（application architecture）是用于实现信息系统的技术规范。

物理数据流图（physical data flow diagram）是一个过程模型，用于交流信息系统的技术实现特征。

分布式系统（distributed system）是一个系统，其构件分布在计算机网络和多个地点。

集中式系统（centralized system）是一个系统，其所有构件都在一个集中的多用户的计算机中。

局域网（Local Area Network, LAN）是一组客户端计算机在相对短的距离内连接到一个或多个服务器。

客户/服务器系统（client/server system）是一种分布式计算方案，其中表现层、表现逻辑层、应用逻辑层、数据处理层和数据层分布在客户端 PC 和一个或多个服务器之间。

瘦客户（thin client）是一台功能不十分强大的个人电脑。

胖客户（fat client）是一台功能更强大的个人电脑、笔记本电脑或工作站。

数据库服务器（database server）是运行一个或多个数据库的服务器。

事务服务器（transaction server）是运行确保所有数据库的修改作为一个整体成功或者失败的服务的服务器。

应用服务器（application server）是运行信息系统的应用逻辑和服务的服务器。

消息或组件服务器（messaging or groupware server）是运行组件服务的服务器。

Web 服务器（Web server）是运行因特网或内联网站点的服务器。

分布式表现（distributed presentation）客户/服务器系统的表现层和表现逻辑层被从遗留系统的服务器上移到客户端上。

分布式数据（distributed data）客户/服务器系统中，数据层和数据处理层放置在服务器上，而应用逻辑层、表现逻辑层和表现层放置在客户端。这也称为两层客户/服务器计算。

分布式数据和应用（distributed data and application）客户/服务器系统中，数据层和数据处理层放置在各自的服务器上，应用逻辑层放置在各自的服务器上，表现逻辑层和表现层放置在客户端上。这也称为三层或 n 层客户/服务器计算。

分割（partitioning）是确定如何在网络中最优地分布或复制应用构件的行为。

网络计算系统（network computing system）是一种多层方案，其中表现层和表现逻辑层在客户端 Web 浏览器中使用从某个 Web 服务器下载的内容实现。

内联网（intranet）是一个使用因特网技术将桌面、工作组和企业计算集成在一起的服务器网络。

分布式关系数据库管理系统（distributed relational database management system）是实现分布式关系数据库的软件。

电子数据交换（Electronic Data Interchange, EDI）是企业之间业务事务或数据的标准化电子流。

软件开发环境（Software Development Environment, SDE）是用于开发信息系统应用的语言和工具包。

水平分层（clean layering）是一种设计策略，它要求表现层、应用层和数据层被物理地分离。

设计单元（design unit）是一个自包含的过程、数据存储和数据流集合，它们共享了类似的设计属性。

12.1 应用架构

第11章介绍了整个系统设计过程的一个高层概述。在系统设计过程的早期，需要开发一个架构蓝图，作为后续的内部和外部设计的提纲。本章主要介绍这个蓝图和应用架构，后续章节将详细介绍每个架构组件的内部和外部设计。架构蓝图将介绍以下设计决策：

- 信息系统的集中或分布程度——大多数现代的系统都分布到网络上，包括内联网和因特网。
- 数据存储在网络上的分布——大多数现代数据库要么是在网络上分布的，要么是在网络上复制的，要么以一种客户/服务器模式出现，要么以一种网络计算模式出现。
- 内部开发的所有软件将使用的实现技术——将使用哪种程序设计语言和工具？
- 商用现成产品的集成——以及对软件的定制需求。
- 用来实现用户界面的技术——包括输入和输出。
- 用来与其他系统接口的技术。

从这些方面考虑来定义信息系统的应用架构。**应用架构**说明用于实现一个或多个（可能所有的）信息系统的技术，作为详细设计、构造和实现的提纲。

在大多数章节中，首先讲授概念和原理，然后再介绍工具和技术。但本章将首先介绍主要工具——物理数据流图。这有两个原因：第一，在第8章中你已经了解了数据流图的系统概念和基本结构；第二，为了介绍不同类型的应用架构，这个工具是一种灵活且相对简单的方法。

尽管将在本章中学习一种新技术——物理数据流图，但这个技术并没有在计算机网络上分解信息系统的架构概念那样重要。

12.2 物理数据流图

在第8章中，数据流图（DFD）是一种系统分析工具，用于建模信息系统的逻辑（即非技术的）业务需求。通过仅仅很少一点图形语言的扩充，DFD也可以用作建模信息系统的物理（即技术的）架构和设计的系统设计工具。**物理数据流图**建模作为信息系统一部分实现的技术设计决策和人为设计决策，将同那些实际构造和实现系统的人沟通技术选择和其他设计决策。换句话说，物理DFD用作系统构造和实现的技术性蓝图。

物理数据流图由Gane、Sarson和DeMarco构思，起初被称为结构化分析和设计的形式化软件工程方法学的一部分。这个方法学特别适合基于大型主机COBOL事务的信息系统和软件，它要求严格详细地说明信息系统的逻辑表示和物理表示。

如今，很少使用上面描述的这种完整的结构化分析和设计方法学，它不适应如今面向对象和基于组件的软件技术，但逻辑的和物理的数据流图技术仍是系统开发的结构化分析和设计时代有用并被大量采用的遗产。

本章将介绍物理DFD的图形符号。物理DFD使用同逻辑DFD（见第8章）一样的基本形状和连接，也就是：a) 过程；b) 外部代理；c) 数据存储；d) 数据流。图12-1显示了一个物理DFD的例子。现在，只要注意到物理DFD所主要显示的技术和实现细节比其逻辑DFD等价更多就可以了。

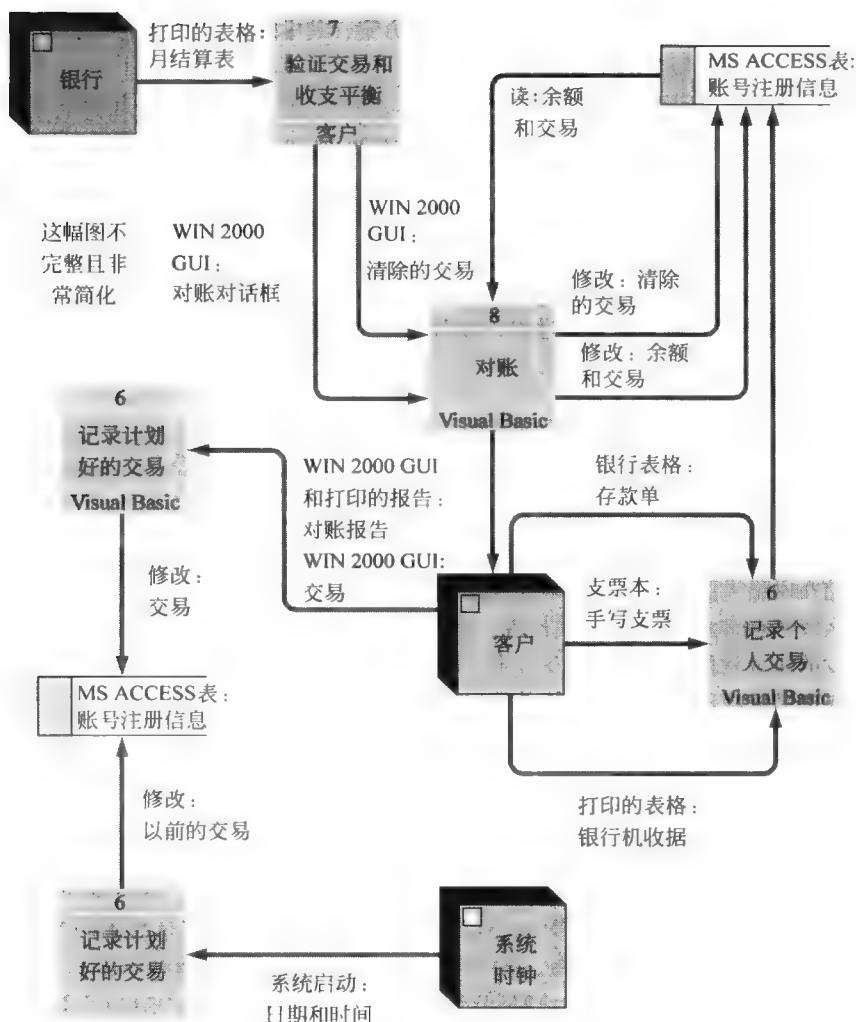


图 12-1 物理数据流图示例

12.2.1 物理过程

过程是 DFD 中的主要形状, 所以 DFD 被称为过程模型。物理 DFD 描述了每个过程计划的实际实现。物理过程是一个处理器 (例如计算机或人), 或者是要执行的特定工作的技术性实现 (例如计算机程序或手工过程)。

在项目较早期 (需求分析过程中), 我们说明了实现基本业务需求所需的逻辑过程, 这些逻辑过程使用逻辑数据流图 (见第 8 章) 建模。现在, 在系统设计过程中, 必须说明如何实际地实现这些逻辑过程。正如上面物理过程的定义中隐含的那样, 物理数据流图有以下两个基本要素:

- 逻辑过程经常被分配到特定的物理处理器, 例如 PC、服务器、大型主机、人或者计算机网络中的其他设备。我们最终可以绘制一个物理 DFD 来建模网络结构。
- 每个逻辑过程必须实现成一个或多个物理过程。注意有些逻辑过程必须分解成多个物理过程, 这是由于以下原因:
 - 为了将过程分解成由人执行的部分和由计算机执行的部分。
 - 为了将过程分解成使用一种技术实现的部分和使用另一种技术实现的部分。
 - 为了显示同一个逻辑过程的多个但又不同的实现 (例如一个过程用于书面订单而另一个过程用于因特网订单)。

■ 为了处理例外，或者为了实现安全需求和审计要求而增加的过程。

在所有这些情况中，如果将一个逻辑过程分解成多个物理过程，或者增加额外的物理过程，为了保持原始数据流的本质不变，就必须相应地增加数据流。换句话说，物理过程仍必须满足逻辑过程需求。

过程 ID 是可选的，但它对匹配物理过程及其逻辑对应物来说是有用的（特别是当逻辑过程要用多个物理过程实现时）。过程名称使用了与第 8 章一样的“行为动词 + 名词/宾语从句”形式，这个名称记录在图 12-2 中，过程实现记录在图形底部。这种记法可能需要根据 CASE 或自动化绘图工具的能力进行调整。图 12-3 的名字演示了同一逻辑过程的各种可能实现。

如果 CASE 工具限制了名称的字符数，你可能必须为使用的实现技术制定并使用一套缩写（可能还要缩写行为动词和宾语从句）。

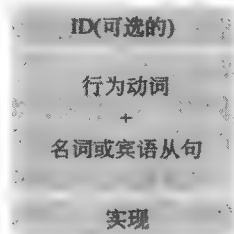


图 12-2 物理过程记法

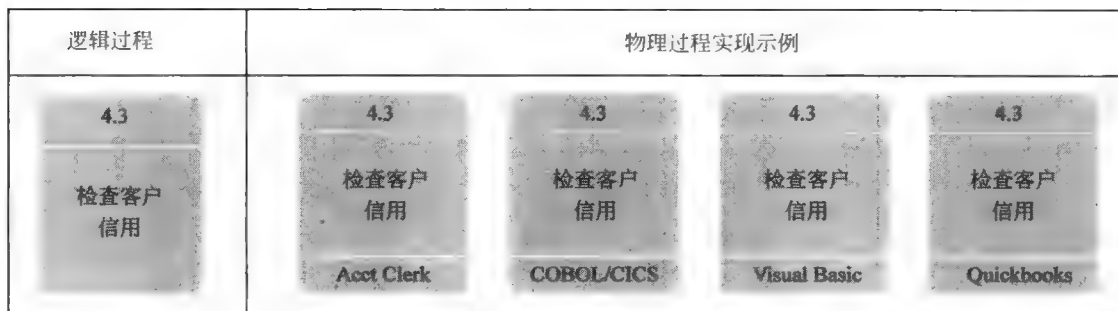


图 12-3 一个逻辑过程的各种可能实现示意图

如果一个逻辑过程一部分由人实现，一部分由软件实现，那么必须把它分解成不同的物理过程，而且物理过程之间必须增加相应的数据流。一个由人（而非由软件）执行的物理过程的名称应该指出谁执行那个过程，建议使用职务或角色，而不要使用真实姓名。图 12-4 是一个例子。

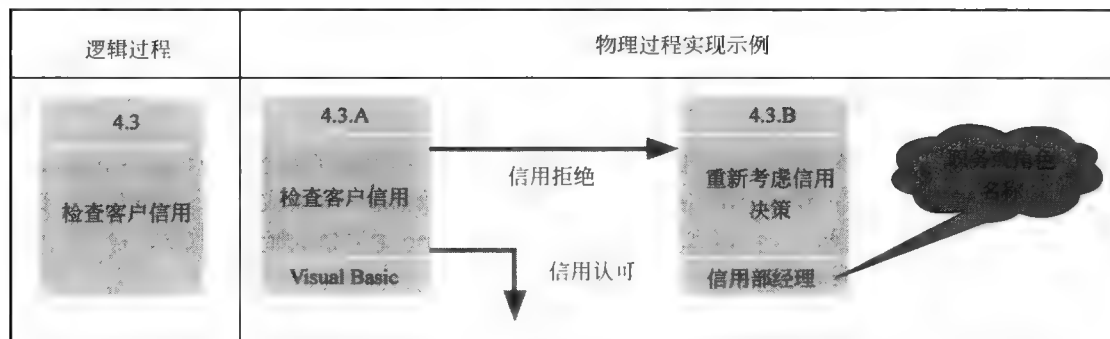


图 12-4 一个逻辑过程的分解示例

我们刚才没有把手工过程（“重新考虑信用决策”）改变成一个外部代理（“信用部经理”），因为整个逻辑过程（“检查客户信用”）是在项目范围内的，所以物理实现的两个方面也都在范围内。直到说明了业务需求的自动化过程和手工过程之后，这个设计才算完整。

对于计算机处理过程，实现方法可以部分地从以下的可能方案中选择一个：

- 购买的应用软件包（例如，Sap [企业软件应用] 或者 Ariba [基于因特网的采购/购买软件应用]）。
- 系统或者工具程序（例如，Microsoft 的 Exchange Server [电子邮件/消息系统] 或者 IBM 的 WebSphere Commerce Business [电子商务框架]）。
- 程序库（简单地用“库”或者库“名称”表示）中的现成应用程序。

- 要编写的程序。一般来说，实现方法说明了用来构造程序的语言或工具，例如：VB、.NET、C++、JAVA、MS ACCESS、PERL 或者 ORACLE DEVELOPER。

还应该介绍最后一个物理过程结构，即多重过程（见图 12-5）。多重过程指示了同样的物理处理器或过程的多个实现。例如，可以使用这个符号来表示多个 PC、多个 PC 上的一个命名程序或者由多个人执行的工作。有些 CASE 工具不支持这种结构，如果不支持，可能需要求助于名称的复数形式来指示一个过程或处理器的多次出现。

许多设计人员喜欢采用一种更实际的表示计算机过程的命名方式，不是使用“名词 + 动词短语”形式，而是使用计算机程序实现的实际源代码文件名。请看图 12-6 中的例子。许多组织都有关于程序名的命名规定和标准。

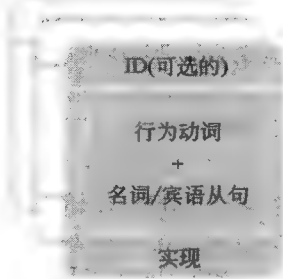


图 12-5 多重过程示意图

逻辑过程	物理过程实现示例	
<div>4.3</div> <div>检查客户信用</div>	<div>4.3.A</div> <div>CHK_CREDIT.COB</div> <div>COBOL+CICS</div>	<div>4.3.B</div> <div>appCheckCredit.vbx</div> <div>Visual Basic</div>

图 12-6 另一种计算机过程命名方式示例

一个物理 DFD 中物理过程的数量几乎总是大于其等价的逻辑 DFD 中逻辑过程的数量，这是因为可能需要增加过程以反映数据的收集、过滤、传递、准备或质量检查——所有这些都在实现视图中。而且，有些逻辑过程可能被分解成多个物理过程，以反映由手工实现和由计算机实现的过程部分，用不同技术实现的过程部分，或者分布到不同的客户端、服务器或主机的过程部分。最后，最重要的是物理 DFD 反映了所选的实现策略需要的所有手工过程和计算机过程。

12.2.2 物理数据流

任何 DFD 中的所有过程都必须至少拥有一个输入数据流和一个输出数据流。物理数据流表示了下列内容：1) 一个物理过程的输入或输出的计划实现；2) 一个数据库的命令或动作，例如创建、读取、修改或删除；3) 通过网络从另一个信息系统输入数据或者向另一个信息系统输出数据；4) 同一个程序中两个模块或子路线之间的数据流。

物理数据流用图 12-7 中模板说明的方式命名。图 12-8 演示了这些命名方法应用于几类物理数据流时的不同情况。

物理 DFD 还必须指出被实现为业务表格的数据流，例如：“表格 23：课程申请”可能是一个学生用来注册课程的单份业务表格。业务表格经常通过拷贝多份（复写纸或不用复写纸）实现。在处理过程的某一点上，不同的拷贝被分拆，并发送到不同的手工过程进行处理，这在物理 DFD 上显示就是一个分叉数据流（见第 8 章）。每个拷贝都应该被唯一地命名，例如：在一个餐馆，客户会收到“表格：信用卡凭证（客户拷贝）”和机器返回的“表格：信用卡凭证（机器拷贝）”。

大多数逻辑数据流将被直接带入物理 DFD 中；有些可能会被合并成表示业务表格的单个数据流；另一些则可能会被分解成多个数据流，作为将逻辑过程分解成多个物理过程的结果；其他一

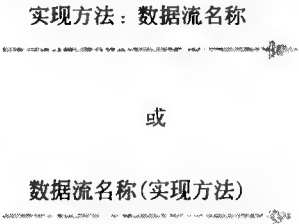


图 12-7 物理数据流命名方式模板

































逻辑数据流	实现	物理数据流例子
—— 订单 → 	计算机输入 (键盘)	— WIN 2000 GUI : — 订单表格 → 
—— 订单 → 	计算机输入 (因特网)	— HTML : — 订单表格 → 
—— 销售的 产品 → 	计算机输入 (无键盘)	— BAR CODE : — 产品UPC → 
—— 工作 时间 → 	计算机输入 (批文件)	— KEY-TO-DISK : — 工作时间 → 
 — 工资分析 →	计算机输出 (打印)	 — PRINTOUT : — 工资分析报告 →
 — 账号历 史记录 →	计算机输出 (联机)	 — WIN 2000 GUI : — 账号历史记录 →
—— 创建 订单 → 	在数据库中创建 一个记录	— SQL Insert : — 新订单 → 
 — 未履行的 订单 →	读取数据库中的记录	 — SQL Select : — 未履行的 订单 →
— 修改信 用度 → 	修改数据库中的 一个记录	— SQL Update : — 信用度 → 
—— 删除雇员 → 	删除数据库中的 一个记录	— SQL Delete : — 雇员 → 
—— 保险事故 索赔 → 	引入一个数据文件	— IMAGE FILE : — 保险事故索赔 → 
 — 课程安排 →	导出一个数据文件	 — Comma Delimited File : — 课程安排 →
 — 增加的 费用 → 	在程序的模块之间 传递数据	 — 增加的 费用 → 
 — 课程 申请 → 	传递一个手工表格	 — Form 23 : — 课程申请 → 

图 12-8 物理数据流示例

些则可能被复制成使用不同技术实现的多个流。例如，逻辑数据流“订单”可能被实现成以下形式：“表格：订单”、“电话：订单”（通过电话接收的口头订单）、“HTML：订单”（通过因特网提交的订单）、“传真：订单”（通过传真接收的订单）以及“消息：订单”（通过电子邮件提交的订单）。

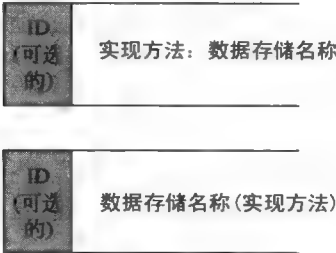
12.2.3 物理外部代理

外部代理从逻辑 DFD 继续不变地转至物理 DFD。为什么？按照定义，外部代理在系统分析期间被归类为系统范围以外，所以，它们不应该变化。只有需求的变化才能激发外部代理的变化。

12.2.4 物理数据存储

在第 8 章中，已经知道了逻辑 DFD 中的每个数据存储表示实体关系图中一个命名实体的所有实例（见第 8 章）。物理数据存储则实现了逻辑数据存储。物理数据存储表示以下内容之一：1）数据库；2）数据库中的表；3）计算机文件（例如 VSAM 或者 B-Tree）；4）重要数据的磁带或介质备份；5）程序需要的临时文件或批处理文件（例如“TAX 表”）；6）任意未经过计算机处理的文件。

当大多数人考虑数据存储时，他们想到的是计算机文件和数据库，但是许多数据存储并没有被计算机化。可以立即想到的例子是文件柜中的书面记录；但是，大多数企业有着更为详细的手工数据存储形式，例如地址卡片、书面目录、各种重要信息和复用信息的便查表格、标准手册、标准操作规程手册、目录等。尽管人们预测了纸质文件的消失，但在可见的将来它们仍将是许多系统的一部分——至少有如下原因：1）对纸有一种心理上的亲切感；2）政府部门经常要求使用纸！



物理数据存储的名称采用图 12-9 显示的格式。图 12-10 显示了物理数据存储名称表示法的一些例子。

逻辑数据存储	实现	物理数据存储
人力资源	一个数据库(多张表)	Oracle: 人力资源DB
市场部	一个数据库视图 (数据库子集)	SQL Server: 东北区市场部DB
采购订单	数据库中的一张表	MS Access: 采购订单
应收账	一个遗留文件	VSAM File: 应收账
税率	静态数据	ARRAY: 税率表
订单	一个脱机归档文件	TAPE Backup: 结束的订单
雇员	一份书面记录文件	File Cabinet: 个人记录
系/人员联系数据	一个目录	Handbook: 系/人员目录
按日期的 课程注册	归档的报告 (供复用和再调用)	REPORT MGR: 课程注册报告

图 12-10 物理数据存储示例

有些设计要求创建临时文件，用作具有不同定时的物理过程之间的队列或者缓冲区。这类文件以同样的方式记录，差别只是其名字应该指示出它们的临时状态。

物理过程、数据流、外部代理和数据存储构成了物理数据流图，这些物理 DFD 建模了信息系统应用中建议（或者计划）的架构，我们最终可以使用那个物理模型来为每个数据存储（见第13章）和每个数据流（见第14章到第16章）设计内部和外部细节。现在，我们已经理解了物理 DFD 的基本组件，下面使用它们介绍一些如今用于信息系统设计的架构。

12.3 信息技术架构

信息技术架构是一个复杂的课题，其本身就值得用一门课程和一本书来介绍。本节将试图总结影响设计决策的现代 IT 技术和趋势。应该注意新的技术在不断地进化，优秀的系统分析员不仅应该更多地了解这些技术，而且还应该理解它们如何工作以及它们的限制。这些细节超出了本书的范围，但应该找到讲解更深入知识的各种技术、数据库和网络课程。系统分析员必须经常阅读流行的行业杂志，与最新的技术保持同步，以保持其客户和信息系统的竞争力。

信息系统框架为理解 IT 架构提供了一个合适的框架。因此，信息系统构件在网络中分布或重复。我们称之为分布式系统架构：

- 架构标准或技术约束在框架的底行显示。注意这些标准或决策要么作为一个独立的架构项目的一部分进行制定（首选的而且应用越来越多），要么作为每个系统开发项目的一部分进行制定。
- 向上指的箭头指出了将影响或者约束设计模型的技术标准。

12.3.1 分布式系统

如今的信息系统不再是基于大型计算机的单一系统。相反，它们构建在网络的某种组合上，以形成分布式系统。分布式系统是一个系统，其中信息系统的构件分布到计算机网络中的多个地点。因此，为支持这些构件所需的处理负载也在网络上的多个计算机之间分布。

分布式系统的对立面就是集中式系统。在集中式系统中，一个集中的多用户计算机（通常是大型主机）运行了信息系统所有构件。用户通过终端（或者目前的 PC 仿真终端）与主计算机交互，但几乎所有的实际处理和工作都在主计算机上进行。

图 12-11 比较了分布式系统架构的各种形式。从概念上说，任何信息系统应用都可以映射到以下 5 层上：

- 表现层是实际的用户界面——对用户输入和输出的表现。
- 表现逻辑层是为了生成表现而必须进行的处理，例如：编辑输入数据和格式化输出数据。
- 应用逻辑层包括支持实际业务应用和规则所需的所有逻辑和处理。例如信用检查、计算、数据分析等。
- 数据处理层包括用来存储和访问往来于数据库的数据所需的所有命令和逻辑。
- 数据层是数据库中实际存储的数据。

图 12-11 中各行显示了这些概念层，列显示了这些形式如何在不同的分布式信息系统架构中实现。从图中可以看出，共有以下三种分布式信息系统架构形式：

- 文件服务器架构。
- 客户/服务器架构。
- 基于因特网的架构。

下面详细地讨论每一种形式。

12.3.1.1 文件服务器架构

如今个人计算机和 workstation 很少被用来支持独立的信息系统，组织需要共享数据和服务，局域网则把许多 PC 和 workstation 连接起来，以互相共享资源并互相通信。局域网（LAN）是一组在相对短的距离内（例如，在一个部门或一个建筑物内）通过电缆或者无线连接到一个或多个服务器（通常是更强大的 PC 或者较大的计算机）的客户端计算机（通常是 PC）。

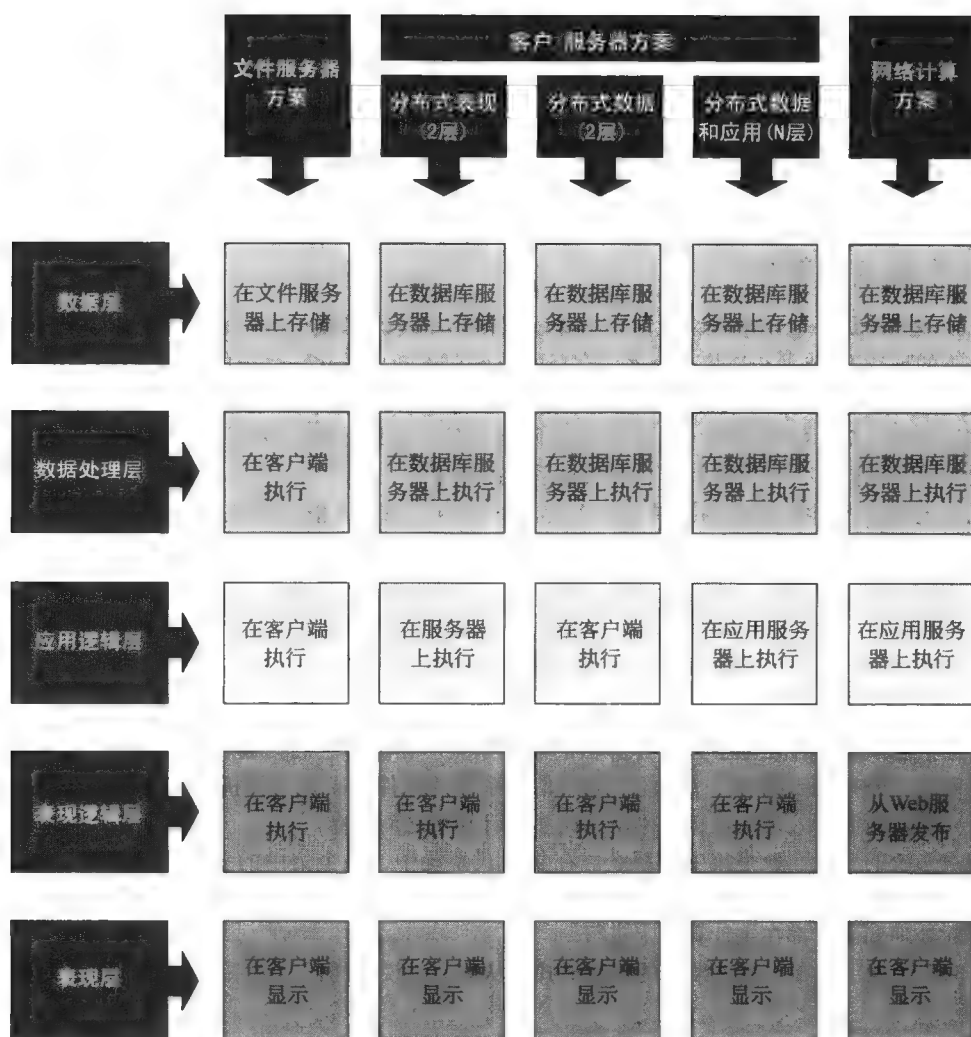


图 12-11 分布式计算和系统的形式

在一个最简单的 LAN 环境中，文件服务器架构被用来实现信息系统。文件服务器系统是一种基于 LAN 的方案，其中服务器计算机仅仅装载了数据层。信息系统应用的所有其他层都在客户端 PC 上实现（注意：文件服务器一般也通过网络共享其他非数据库文件——例如字处理文档、电子表格、图像和图形、工程图、演示汇报等）。图 12-12 演示了一个文件服务器架构，这个架构是许多 PC 数据库引擎（例如 Access 和 FoxPro）使用的典型结构（想一想！虽然你的 Access 数据库可以在一个网络服务器上存储，但是实际的 Access 程序必须安装到使用数据库的每个 PC 上，并从那里执行）。

文件服务器架构仅仅对于共享用户数相对较少的小型数据库应用来说是可行的，因为如果应用仅仅想检查数据库中的一个记录（例如一个客户），记录的整个文件或表都必须首先被下载到执行数据处理逻辑的客户端 PC 上，然后才能读取想要的记录。

12.3.1.2 客户/服务器架构

当前盛行的分布式计算模型叫做客户/服务器计算（尽管它正快速地让位于基于因特网的模型）。客户/服务器系统是一种方案，其中表现层、表现逻辑层、应用逻辑层、数据处理层和数据层分布在客户端 PC 和一个或多个服务器之间。

客户计算机可以是个人计算机、工作站、“有时连接的”笔记本电脑、掌上电脑（例如，手持式或者 Windows CE 平台）、Web TV 或者任何带有嵌入式处理器并可以连接到网络上的设备（例如一个生产

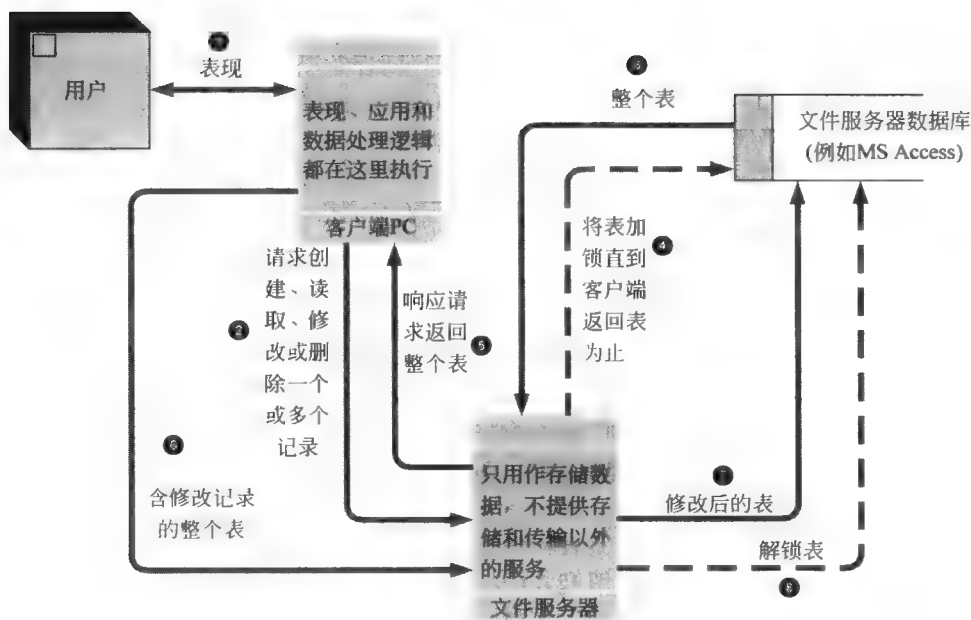


图 12-12 文件服务器架构

车间的机器人或控制器)的任何组合。客户端可以是瘦客户,也可以是胖客户。瘦客户是一台在处理器速度和内存方面功能不十分强大的个人电脑,它仅仅给用户表现界面(屏幕)。换句话说,它仅作为一个终端使用,例如 Windows 终端和 X/Windows。在瘦客户计算中,实际的应用逻辑在远程的应用服务器上执行(例如 Citrix 服务器或者 Microsoft Windows 终端服务器)。胖客户一般是一台在处理器速度、内存和存储容量方面功能更强大的个人电脑、笔记本电脑或工作站。几乎所有的 PC 都被认为是胖客户。

客户/服务器模型中的服务器必须比文件服务器模型中的服务器功能更强大。实际上,大型主机可以在客户/服务器方案中充当服务器的角色,但更典型的方式是运行具有客户/服务器能力的操作系统(例如 UNIX、Windows Server 2003、Linux)的网络服务器。有好几类服务器可以用于客户/服务器方案中,这些服务器可以驻留在独立的物理服务器上,或者被合并到较少的服务器中。

- **数据库服务器**运行一个或多个共享的数据库(像文件服务器一样),还执行信息系统的所有数据库命令和服务(同文件服务器不同)。大多数数据库服务器运行某个 SQL 数据库引擎,例如 Oracle、Microsoft SQL Server 或者 IBM DB2 Universal Database。
- **事务服务器**运行最终确保所有单个业务事务数据库的修改作为一个整体成功或者失败的服务,例如 IBM CICS 和 BEA。
- **应用服务器**运行信息系统的应用逻辑和服务。它必须同用于表现的前台客户端通信,并同用于数据访问和修改的后台数据库服务器通信。应用服务器经常与事务服务器集成在一起。大多数应用服务器以 CORBA 对象共享标准或者 Microsoft COM+ 标准为基础。
- **消息或组件服务器**运行电子邮件、日历和其他工作组服务,这类功能可以被实际地集成到信息系统应用中,例如 Lotus Notes 和 Microsoft Exchange Server。
- **Web 服务器**运行因特网或内联网站点,通过向胖客户和瘦客户返回文档(例如以 HTML 格式)和数据(例如以 XML 格式)同它们进行通信。一些 Web 服务器被专门地设计成运行电子商务应用(例如 IBM 的 WebSphere Commerce Business)。

客户/服务器架构本身就具有几种形式,每种形式都需要详细解释,图 12-11 比较了这些 C/S 形式。

12.3.1.3 客户/服务器——分布式表现

大多数集中式(或大型主机)计算应用使用一种老式的字符用户界面(CUI),它与如今的图形用

户界面 (GUI) (例如 Microsoft Windows 和 UNIX X/Windows) 比较起来就显得既麻烦又笨拙 (更不用提 Web 浏览器, 例如 Netscape Navigator 和 Internet Explorer)。随着个人计算机快速地替代哑终端, 用户越来越感觉这种 GUI 技术好用。而且随着用户对 PC 工具 (例如字处理和电子表格软件) 越来越熟悉, 他们就会希望集中式遗留计算应用也能具有类似使用 GUI 的感觉。

那么就使用分布式表现系统吧! 在分布式表现客户/服务器系统中, 表现层和表现逻辑层被从遗留系统的服务器上移到客户端上, 应用逻辑层、数据处理层和数据层仍保留在服务器上 (通常是一台大型主机)。这种结构有时称为“穷人”的客户/服务器, 它构建在集中式计算应用之上, 并增强了集中式计算应用。从本质上说, 旧式的 CUI 被从遗留应用系统中剥除, 然后重新生成将运行在 PC 上的 GUI。换句话说, 只有用户界面 (或者表现层) 被分布到了客户端。

分布式表现具有以下几个优点: 第一, 它实现起来相对较快, 因为遗留应用系统的大部分仍保持不变; 第二, 用户得到一个到遗留系统的快速、友好和熟悉的用户界面, 至少看上去有点类似于 PC 工具的界面; 最后, 遗留系统的有效生命期可以得到延长, 直到资源足够进行应用系统的整体重新开发为止。缺点是应用系统的功能得不到明显地提高, 而且由于只处理了用户界面, 所以这个方案也没有充分利用客户端的桌面计算机潜力。

有一类 CASE 工具 (有时称为屏幕生成器) 自动地读取 CUI, 并生成一个可以通过 GUI 编辑器修改的初始 GUI。图 12-13 演示了这个技术。图 12-14 显示了分布式表现方案的物理 DFD。

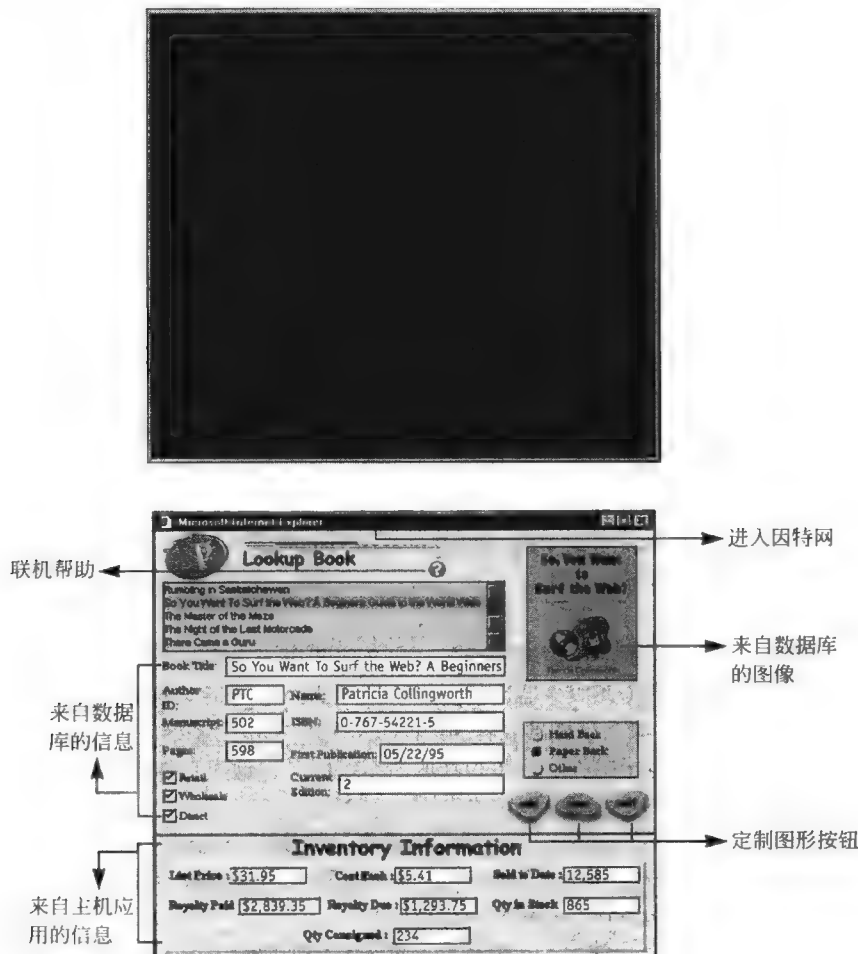


图 12-13 从 CUI 构造 GUI

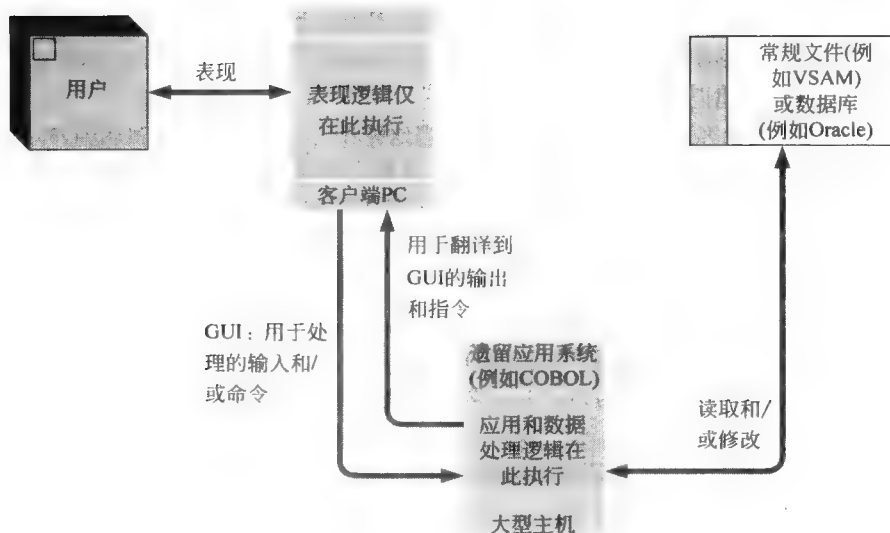


图 12-14 客户/服务器系统：分布式表现

12.3.1.4 客户/服务器——分布式数据

这是真正的客户/服务器计算的一种最简单形式，其中局域网通常将客户端连接到服务器上。在分布式数据客户/服务器系统中，数据层和数据处理层放置在服务器上，而应用逻辑层、表现逻辑层和表现层放置在客户端。这也称为两层客户/服务器计算。图 12-15 演示了两层的分布式数据客户/服务器系统的物理 DFD。

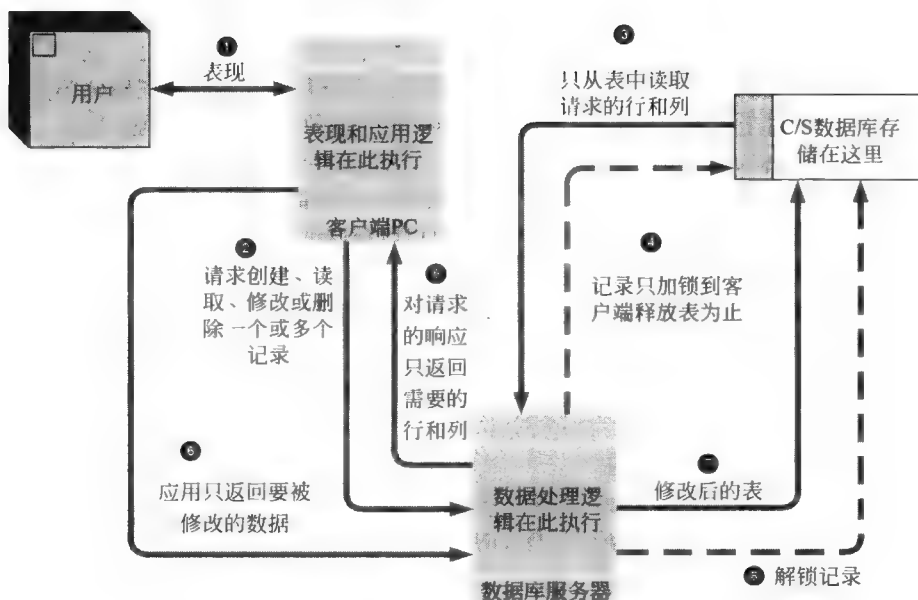


图 12-15 客户/服务器系统：分布式数据（两层）

理解文件服务器系统和分布式数据客户/服务器系统的区别很重要。它们都在服务器上存储实际数据库，但只有客户/服务器系统在服务器上执行所有的数据处理命令（例如用于创建、读取、修改和删除记录的 SQL 指令）；在文件服务器系统中，数据处理命令必须在客户端实现。与文件服务器方案相比，分布式数据客户/服务器方案具有以下优点：

- 网络流量少很多，因为只有数据库请求和需要的数据库记录实际地在客户端工作站之间来回传输。
- 数据库完整性更容易维护。一般只需锁定客户端使用的记录，其他客户端可以同时使用同一个表或数据库中的其他记录。

为了处理应用逻辑层，客户端工作站仍然必须相当健壮（“胖”）。应用逻辑通常用一种客户/服务器编程语言编写，例如 Sybase 公司的 PowerBuilder、微软公司的 Visual Basic .NET 或 C#。这些程序必须为客户端编译并在客户端执行。为了提高应用系统的效率并减少网络流量，有些业务逻辑可以以存储过程（在下一章讨论）的形式分布到数据库服务器上。

数据库服务器是这个架构的基础。数据库服务器存储了数据库，但它们也直接在服务器上执行数据库指令。客户端仅仅发送数据库指令到服务器，服务器仅仅返回数据库命令处理的结果——并非完整的数据库或表，所有的高端数据库引擎（例如 Oracle 和 SQL Server）都使用这种方法。分布式数据架构可能包含多个数据库服务器，数据可以在几个数据库服务器之间分布，或者在几个数据库服务器上复制。

两层客户/服务器主要的缺点是应用逻辑必须在所有客户端上复制和维护，这可能有几百个或者几千个客户端。设计人员必须为版本升级做计划，提供控制以确保每个客户端都运行业务逻辑的最新发布版，并确保 PC 上的其他软件（购买的或者内部开发的）不会干扰业务逻辑。

12.3.1.5 客户/服务器——分布式数据和应用

当客户端的数量增长时，两层系统经常产生性能问题，这些性能问题与在客户端上执行所有业务逻辑的低效率有关。而且，在一个多用户的事务处理系统中（也称为联机应用处理或 OLAP），必须软件管理事务，以确保与事务相关的所有数据作为一个整体进行处理，这通常要求采用多层客户/服务器方式的分布。在分布式数据和应用客户/服务器系统中：1）数据层和数据处理层放置在各自的服务器上；2）应用逻辑层放置在各自的服务器上；3）只有表现逻辑层和表现层放置在客户端上。这也称为三层或 n 层客户/服务器计算。

三层客户/服务器方案使用了与两层客户/服务器方案中同样的数据库服务器。另外，三层系统引入了一个应用和/或事务服务器。通过将应用逻辑移到自己的服务器上，应用逻辑只需在服务器上维护，而无需在所有客户端上维护。图 12-16 描绘了这种三层方案的一个物理数据流程图。

三层客户/服务器逻辑可以使用微软的 Visual Basic .NET 或 C# 的语言组合一个事务监视器编写，并分配到多个服务器上。高端工具（例如 Forté）提供了更强大的功能，可以在复杂的网络中分布应用逻辑和数据。与数据库服务器方案一样，有些业务逻辑可以以存储过程的形式分布到数据库服务器上。

在一个三层系统中，客户端执行整个系统组件中最小的部分，只有用户界面和一些相对稳定的或个人的应用逻辑需要在客户端执行，这简化了客户端的配置和管理。

三层客户/服务器最大的缺点是设计和开发上的复杂性，其设计中最困难的部分是分割。分割是确定如何在网络中最优地分布或复制应用构件的行为。好在 CASE 工具正稳步地改进，可以提供对分割功能的更多支持。

12.3.1.6 基于因特网的计算架构

有些人将基于因特网的系统架构简单地看成是客户/服务器所取得的最新进展。在本节中，我们将基于因特网的计算架构看作是一种具有不同形式的分布式架构，它正快速地重新塑造系统分析和信息技术的设计思考过程。

网络计算系统是一种多层方案，其中表现层和表现逻辑层在客户端 Web 浏览器中使用从某个 Web 服务器下载的内容实现，表现逻辑层然后连接到运行在应用服务器上的应用逻辑层，它最终连接到后台的数据库服务器。假设，所有的信息系统都在浏览器中运行（财务系统、人力资源系统、生产系统）所有的系统。电子商务是其中的一部分，当本书出版时，电子商务应用正在获得广泛的重视。同样用于构造电子商务方案的因特网技术正在被用于重新塑造大部分企业的内部信息系统——我们称之为电子业务（尽管这个词也有多种解释）。按照我们的观点，网络计算本质上与我们描述的客户/服务器系统是不同的。

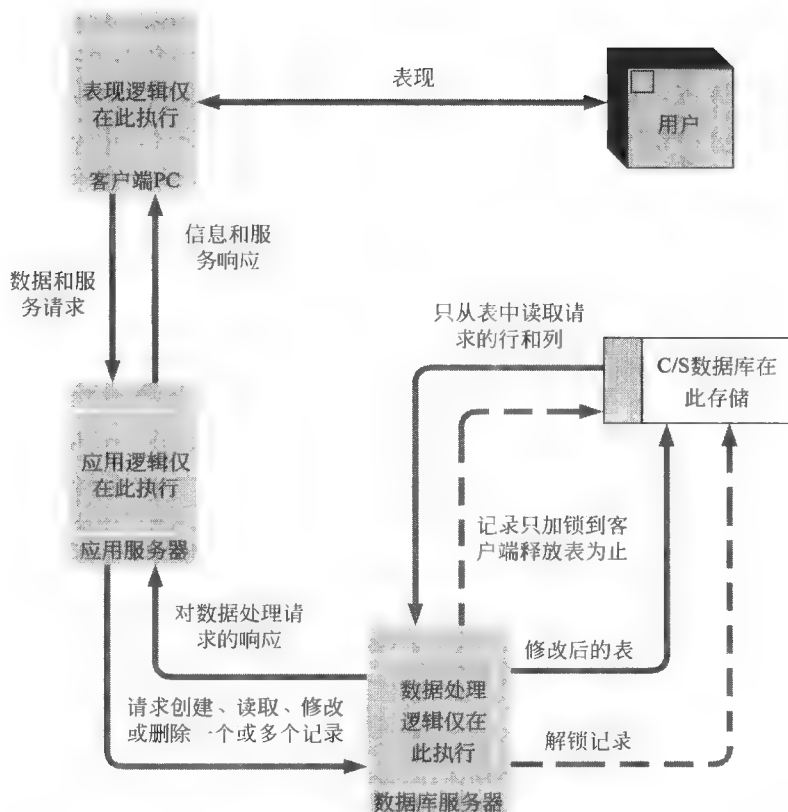


图 12-16 客户/服务器系统：分布式数据和应用（三层）

很少有新技术能够像因特网或万维网这样在企业和社会上如此迅速地增长。因特网扩展了信息系统和事务处理系统的范围，把潜在客户、客户、合作伙伴、远程客户、供应商、政府甚至竞争对手都包括了进来。在 20 世纪 90 年代后期，因特网仍主要被用于在虚拟市场上树立企业形象，发布关于产品和服务的公共信息，以及为以客户为中心的服务提供新基础。但如今，大多数企业都十分关注开发允许客户直接在 Web 上交互和进行交易的电子商务方案（例如直接面向客户的商店）。我们甚至看到了虚拟企业的出现，也就是完全在 Web 上“进行交易”的企业（例如，Amazon.com [图书和音像]、ETrade [股票和期货]、eBay [拍卖] 和 Buy.com [电器和用具]）。最引人注目的争论之一是这些“click-and-mortar”虚拟公司是否可以赢利，并同更传统的“brick-and-mortar”公司真正地竞争，后者正以多样化的方式快速进入这个虚拟市场。

但是这种因特网技术的最大潜力实际上可能在于它应用于内联网上的传统信息系统应用和开发方面。内联网是一个安全的网络（通常是公司内部网络），它使用因特网技术将桌面、工作组和企业计算集成为一个统一的系统。每个应用都在（或至少从）浏览器中运行——日常应用（例如字处理和电子表格）；工作需要的任何（以及所有）传统信息系统应用（例如，财务、采购、人力资源等）；所有的电子邮件、日历和工作组服务（例如，虚拟会议和小组编辑文档）；当然还有所有与工作有关的外部因特网链接。

这个概念应该不难把握。每个雇员的“起始页”都是进入完成其自己的全部工作需要的所有信息系统和服务的门户。因为所有的东西都在 Web 浏览器上运行，所以不用再担心存在多个不同的计算机架构（Intel、Motorola 与 RISC）或者为多个不同的计算机架构开发应用，也不用再担心存在不同的桌面操作系统。图 12-17 显示了网络计算的一个物理数据流图，注意 Web 服务器被添加到前面的三层模型中，这个 DFD 还显示了网络计算的电子商务（企业到客户）和电子业务（企业到企业）方面的内容。

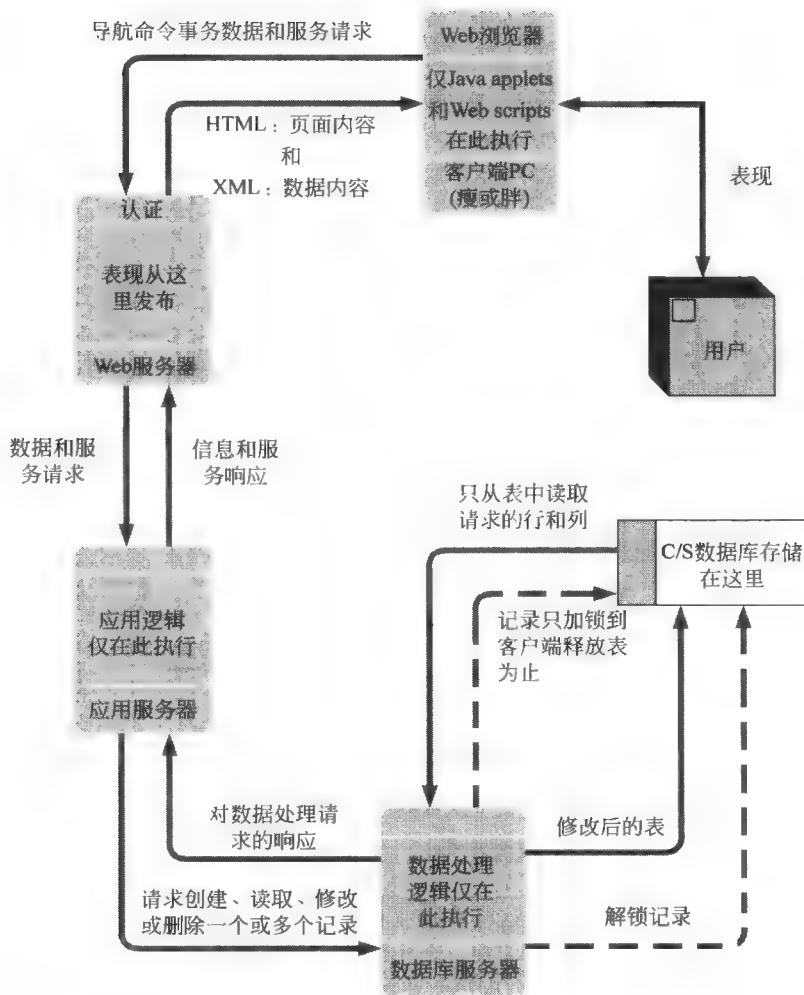


图 12-17 网络计算系统：因特网/内联网

12.3.2 数据架构——分布式关系数据库

客户/服务器和网络计算已经使得分布数据而不失去控制成为可能，这种机制是通过分布式关系数据库技术的进步实现的。关系数据库是以表的形式存储数据。每个文件实现成一张表，每个域是表中一列，文件中的每个记录是表中一行。两个表（例如，“客户”和“订单”）之间相关的记录通过在两个表之间内部地复制列来实现（在这个例子中，“客户编号”既存储在“客户”表中，也存储在“订单”表中）。分布式关系数据库分布或者复制表到位于重要地理位置的多个数据库服务器上（例如不同的销售区）。实现分布式关系数据库所需的软件被称为分布式关系数据库管理系统。分布式关系数据库管理系统（或者分布式 RDBMS）是一个软件程序，它控制到以关系格式存储的数据的访问和维护，还提供了备份、恢复和安全功能。它有时被称为客户/服务器数据库管理系统。

在一个分布式 RDBMS 中，处理所有数据库命令的底层数据库引擎在数据库服务器上执行。其优点是减少了网络上的数据流量，除了最小的系统（以用户数量度量）以外，对于所有系统来说这都是一个显著的优点。分布式关系 DBMS 还提供了更复杂的备份、恢复、安全、完整性和处理功能（虽然随着每个 PC RDBMS 新版本的发布，分布式 RDBMS 与 PC RDBMS 的差异正变得越来越小）。

分布式 RDBMS 的例子包括 Oracle 公司的 Oracle、IBM 公司的 Universal Database 系列、微软公司的 SQL Server 和 Sybase 公司的 Sybase。大多数 RDBMS 支持两类分布式数据。

- 数据分割实际地分布行和列到特定的数据库服务器上，服务器之间很少或者没有重复。不同的列可能被分配到不同的数据库服务器上（称为垂直分割），或者表中不同的行可能被分布到不同的数据库服务器上（称为水平分割）。
- 数据复制在多个数据库服务器上复制一些表或所有表（行和列）。整个表可以被复制到某些数据库服务器上，同时表中行的子集可以被复制到其他数据库服务器上。具有复制技术的 RDBMS 不仅控制每个数据库服务器上数据库的访问和管理，而且传播数据库服务器上的修改到数据被复制到的其他数据库服务器上。

对于一个给定的信息系统应用，“数据”架构必须说明 RDBMS 技术，以及数据将被分割或复制的程度。记录这些决策的一种方法是按图 12-18 的方式把它们记录在物理数据存储中，注意如何使用 ID 域来指示分割（P）和复制（主拷贝用 M、复制拷贝用 R）的代码。在前一种情况下，应该说明哪些行和（或）列被分割到物理数据库中。

逻辑数据存储	使用分割技术的物理数据存储	使用复制技术的物理数据存储
<div>1</div> <div>CUSTOMERS</div>	<div>1P#</div> <div>Oracle 7: REGION 1 CUSTOMERS</div> <div>1P#</div> <div>Oracle 7: REGION 2 CUSTOMERS</div>	不可应用。办事处不需要访问销售区以外的客户数据
<div>2</div> <div>PRODUCTS</div>	不可应用。所有办事处都需要访问所有产品(无论哪个销售区)的数据	<div>2M</div> <div>Oracle 8i: PRODUCTS (Master)</div> <div>2R</div> <div>Oracle 8i: PRODUCTS (Replicated Copy)</div>

图 12-18 示意图

应用的“数据”架构根据期望的客户/服务器或者网络计算模型以及支持那个模型所需的数据库技术来选择。许多组织已经标准化了供选择的 PC RDBMS 以及首选的分布式企业 RDBMS，例如音阶公司已经标准化了微软公司的 Access 和 SQL Server。通常，在任何关于要使用的数据库技术以及将使用那个技术的任何数据库设计的讨论中都应该包括一个资深数据库管理员。

12.3.3 接口架构——输入、输出和中间件

必须做出决策的另一项基本信息技术涉及输入、输出和系统间连接。这个决策曾经很简单——批处理输入或者联机输入；但如今还必须考虑各种现代技术，例如自动识别、笔数据输入、各种图形用户界面、电子数据交换、图像、语音识别等。下面简要地介绍这些技术及其物理 DFD 结构。

12.3.3.1 批处理输入或输出

在批处理中，事务积聚成批进行周期性的处理。系统处理批处理输入，修改数据库并产生相应的输出，大多数输出按照进度安排生成到纸或微缩胶片上，其他输出则可以按需要产生或者在一个特定的时间段内产生（例如 24 小时）。

与常见的看法不同，批处理输入技术并没有完全过时。虽然我们如今很少看到穿孔卡片和磁带批处理了，但有些应用需求仍要求使用批处理：也许输入自然地成批到达（例如邮件），也许输出自然地成批产生（例如发票）。许多组织仍然成批地收集和处理计时卡片。但是，从批处理输入向联机输入的发展趋势是确定不变的。同时，key-to-disk 文件最常用，它的物理数据流结构如图 12-19 所示。首先，注意逻辑名称是单数，但批数据名称为复数。另外，注意批数据进入一个临时的数据存储，之后被一个按日期触发的“发薪水”过程读取。

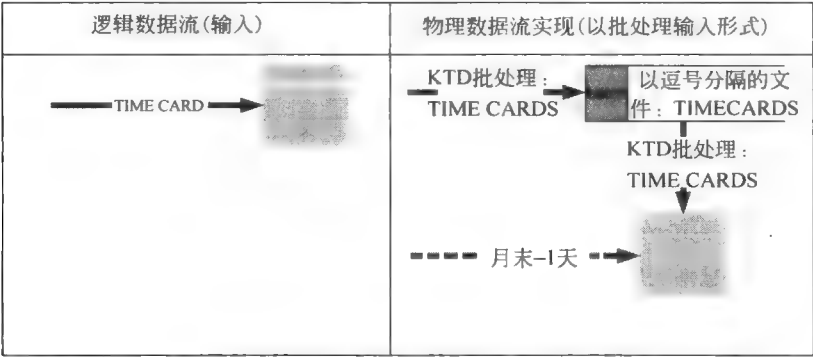


图 12-19 key-to-disk 文件的物理数据流结构

批处理输出则完全不同。许多应用使用批处理输出，例如生成发票、账号结算、成绩报告、付薪水的支票、W-2 税收表格等。批处理输出经常具有一个公共的物理特征——使用预先打印的表格。对你来说，应该不难想象出一个预先打印的表格被加载到打印机上，然后生成前面提到的例子输出。物理数据流结构看上去将有点像图 12-20 中的结构，注意复数名称反映了批处理。

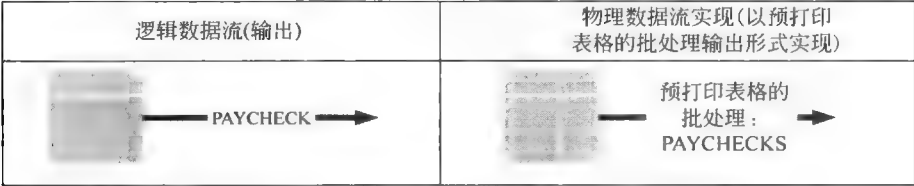


图 12-20 以批处理输出形式实现的物理数据流结构

当老式的批处理系统要遭到淘汰时，我们应该探索其他的物理实现技术。

12.3.3.2 联机输入和输出

大多数系统已经慢慢地从批处理进化到联机或实时处理。联机输入和输出在用户和计算机应用之间提供了更具会话性的对话过程，它们还提供了接近即时反馈，以响应事务、问题和查询。在如今快速发展的经济中，大多数企业事务和查询最好尽可能快地处理。因为数据录入和输入之间没有时间间隔（同在批处理中的情况比较），所以错误能得到更快地确定和改正。此外，联机方法允许决策中有更多的交互。

如今大多数系统正被设计成联机处理系统，即使数据以自然成批的方式到达也是如此。从技术上讲，所有的 GUI 和 Web 应用都是联机的或实时的，而且因为已经了解到那些架构是客户/服务器和网络计算中的首选，所以可以预期大部分物理数据流将用某种类型的 GUI 技术实现。物理数据流结构类似于图 12-21 中的形式。对于物理输出，注意物理输出的两种格式都是可能的。我们将增加连接符号（见第 8 章），以使流互斥地与（都需要），或者互斥地或（或者，但不都是）。

12.3.3.3 远程批处理

远程批处理组合了批处理以及联机输入和输出两者的优点。分布式联机计算机处理数据输入和编辑，被编辑的事务收集到一个批处理文件中供以后传输到主计算机，主计算机以批处理形式处理文件，结果通常作为一批传输回原来的计算机。

远程批处理不算是一种新技术，但是个人计算机给了这种技术赋予新的生命。例如，作者的一个同事每学期使用微软的 Access 程序为系里输入并测试一个课程表，当完成后，他生成一个用逗号分隔的文件，并传输给课程调度部门进行批处理，整个输入模型看上去类似于图 12-22 的情况。

随着掌上电脑和小笔记本电脑技术的发展，使用 PC 的远程批处理应该会获得另一次发展机会。这些 4 盎司到 4 磅重的计算机可以用来收集批信息，包括从库存数量到抵押应用的每样数据。输入在设备上远程地组织成批，供以后作为一批信息进行传输。

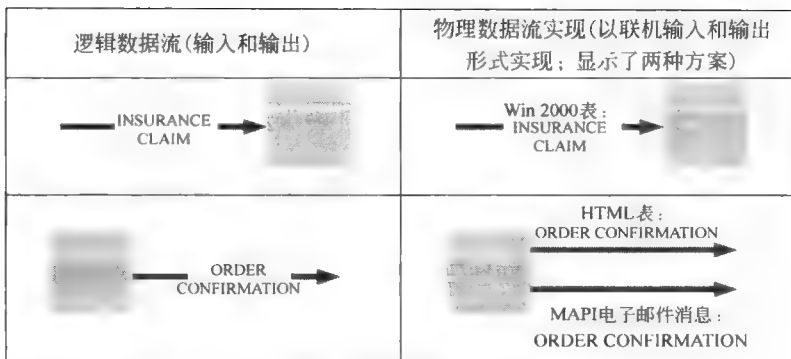


图 12-21 联机处理系统的物理数据流结构

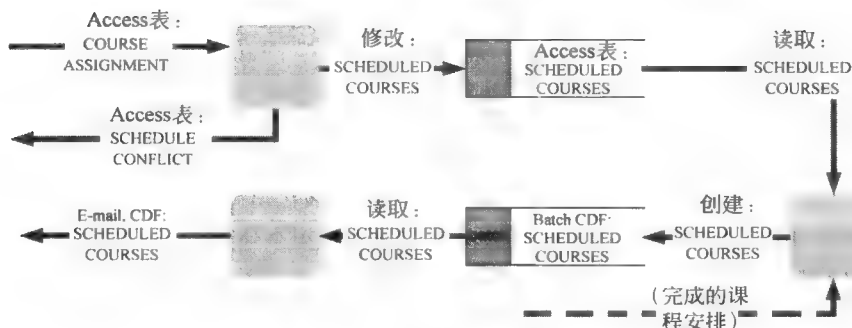


图 12-22 远程批处理输入模型

12.3.3.4 无键盘数据输入 (和自动识别)

键入数据一般是计算机输入 (和查询) 中的主要错误来源, 所以在系统设计中应该考虑任何可以减少或者消除键入错误的技术。在批处理系统中, 键入错误可以通过光字符阅读 (OCR) 和光标记阅读 (OMR) 技术消除, 这两种技术在输入设计中都是可行的, 其物理数据流结构如图 12-23 所示。

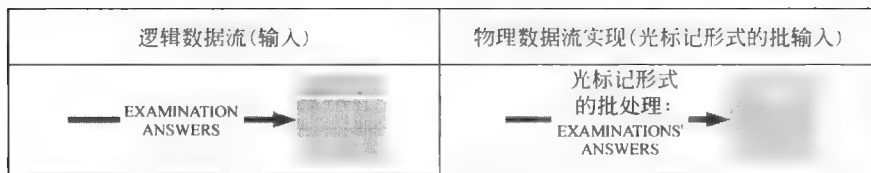


图 12-23 光标记形式的批输入的物理数据流结构

无键盘数据输入以自动识别系统的形式进入联机系统。例如, 条形码技术 (例如在零售业中常用的统一产品代码) 广泛地应用于许多现代应用中。再比如, 当你把包裹送到一个投递中心投递时, 联邦快递公司所有的包裹创建一个条形码标签。包裹被运送到其最后目的地的过程中, 条形码可以被阅读和跟踪。为了在更小的标签里压入更大的信息量, 条形码技术正在不断地改进。无键盘数据输入的物理数据流结构如图 12-24 所示 (接收物理过程将用它执行的功能命名)。

12.3.3.5 笔输入

随着基于笔的操作系统 (例如, Palm OS 和微软公司的 Windows Mobile) 的广泛应用, 以及构造基于笔的应用系统的工具标准化, 可以预期将有更多使用这项技术的系统设计。

有些企业已经使用了这项技术进行远程数据收集。例如, UPS 公司使用基于笔的笔记本系统帮助跟踪投递系统的包裹: 司机在特殊的表格计算机上调出包裹跟踪编号, 客户在指定的区域签名, 当司

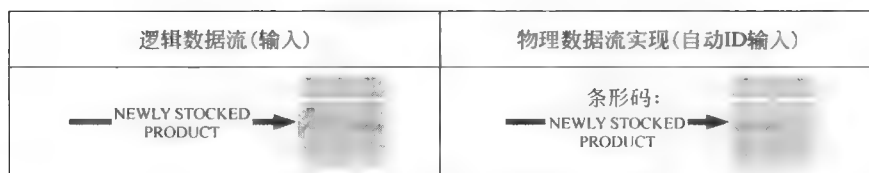


图 12-24 自动 ID 输入的物理数据流结构

机回到卡车上并把表格计算机放回它的扩展坞时,修改后的投递数据通过移动电话调制解调器传送回投递中心,在投递中心包裹跟踪系统修改数据库(最终发货人只需通过一次简单的 Web 查询就可知道包裹已经收到了)。如图 12-25 所示。

12.3.3.6 电子消息和工作组技术

电子邮件已经成熟。它不再仅仅是一种比较有效的沟通方式,人们正在设计直接包含这项技术的信息系统。例如,微软公司的 Exchange Server 和 IBM 公司的 Lotus Notes 能够构造可以集成到任何应用中的智能电子表格。基本的消息服务也可以被集成到应用中。

下面是一个例子:雇员通过一个基于电子邮件的表格可以发出出差请求。根据表格中提交的数据,预定义的规则可以自动将请求转送到相应的决策人。例如,费用不高的出差请求可以直接转发到业务办公室;费用较高的请求可以首先转发到部门领导批准,然后再到业务办公室。最后,被批准的表格可以自动地输入到相应的报销处理系统进行处理。而在每一步中,消息系统自动通过电子邮件通知申请人进展情况。涉及一个电子邮件消息实现的物理 DFD 前面已经介绍过。

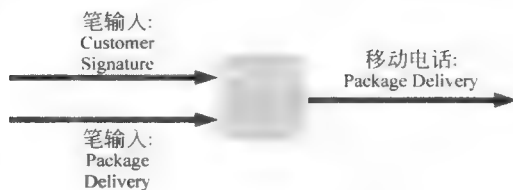


图 12-25 UPS 公司远程数据收集中心笔输入技术的应用

12.3.3.7 电子数据交换

有两类企业经常使用电子数据交换:有许多运营地点的企业和寻求更有效地与其他企业交换事务和数据的企业。电子数据交换(EDI)是企业之间业务事务或数据的标准化电子流。一般来说,许多企业必须采用一种使 EDI 可行的数据格式。

使用 EDI,企业可以消除对书面文档和邮件的依赖性。例如,现在大多数大学接受来自国家测试中心通过 EDI 发送的 SAT 或 ACT 测试成绩,这是可以实现的,因为大学注册人员已经就用于这些测试成绩的标准格式达成了一致。如图 12-26 所示。

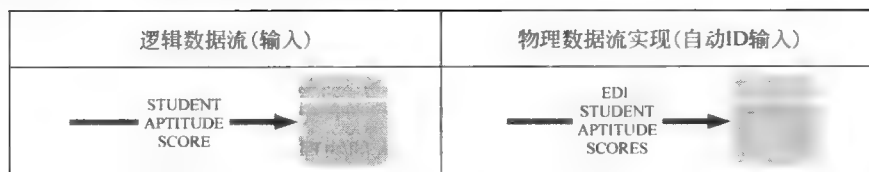


图 12-26 使用电子数据交换的大学测试成绩发送系统的物理数据流结构

12.3.3.8 图像和文档交换

另一种新兴的 I/O 技术是图像和文档交换,它类似于 EDI,差别在于它传输和接收表格和数据的实际图像。图像和文档交换对于需要表格图像或图形的应用特别有用,例如保险行业在电子化传输、存储和使用索赔图像的应用方面卓有成效。其他图像应用组合了数据和图片或图形,例如,法律仲裁应用可以存储、传输和接收照片和指纹图像。

12.3.3.9 中间件

以上内容大部分都集中在输入和输出(即用户界面)的讨论上,但许多系统设计要求使用过程到过程的物理数据流。在本章前面,描述了各种客户/服务器和网络计算结构,这些结构中自动地包含了

过程到过程的数据流，因为客户端和服务端之间必须会话，而它们通过中间件实现会话。中间件是支持系统中不同处理器之间通信的工具软件，可以在相应的操作系统中构造它，或者通过购买的中间件产品加入。中间件产品使得程序员可以忽略底层通信协议。

把中间件比喻成是客户/服务器中的“黏合剂”。有三类中间件，它们恰好对应分布式系统框架的中间三层：表现逻辑层、应用逻辑层和数据处理逻辑层。

- 表现中间件使得程序员可以构建能够与 Web 浏览器或桌面 GUI 交互的用户界面组件。例如，HTTP 使得程序员可以通过一个标准的应用程序接口（API）与 Web 浏览器通信。
- 应用中间件使得程序员编写的在不同处理器上运行的两个过程以最合适的方式互相通信（对整个应用来说）。应用中间件是多层应用开发的基础，其例子有远程过程调用（RPC）、消息队列和对象请求代理。
- 数据库中间件使得程序员可以通过一个标准 API 传递 SQL 命令到数据库引擎进行处理。

另一类常见的中间件是 ODBC（开放数据库互连）和 JDBC（Java 数据库互连），它们自动地将对一个数据库服务器使用的 SQL 命令转换成在不同的数据库服务器上使用的命令（例如，Oracle 到 SQL Server，或者反过来）。

在物理数据流图中，中间件可以通过在物理数据流中说明中间件类名（例如 ODBC）进行描述。

12.3.4 过程架构——软件开发环境

一个应用的“过程”架构按照将用来开发这个过程的业务逻辑和应用程序的软件语言和工具进行定义。通常，这可以表述成一个可供选择的菜单，因为不同的软件开发环境适合于不同的应用。软件开发环境（SDE）是用于开发信息系统应用的语言和工具包。对 SDE 进行分类的一种方法是按照它们支持的客户/服务器或者网络计算架构的类型进行。

12.3.4.1 用于集中式计算和分布式表现的 SDE

以往用于集中式计算的 SDE 很简单，它包括以下内容：

- 一个编辑器和编译器（通常是 COBOL），用于编写程序。
- 一个事务监视器（通常是 CICS），用于管理联机事务和终端屏幕。
- 一个文件管理系统（例如 VSAM）或者数据库管理系统（例如 DB2），用于管理存储的数据。

这就是全部内容。因为所有这些工具都在大型主机上运行，所以只有计算机操作系统（绝大多数情况下是 MVS）最重要。

个人计算机把许多新的 COBOL 开发工具带到了大型主机领域。基于 PC 的 COBOL SDE（例如 Micro Focus 的 COBOL Workbench）通常给程序员提供了更强大的工作站级编辑器、测试和调试工具。程序员可以在这个层次上进行很多开发工作，然后把代码上载到中心计算机进行系统测试、性能调试和代码生成。通常，SDE 将与 CASE 工具和代码生成器结合，充分利用系统分析阶段开发的过程模型。

最后，SDE 提供了开发分布式表现客户/服务器系统的工具。例如，Micro Focus 的 Dialog Manager 向 COBOL Workbench 用户提供了构建可以与 CICS 事务监视器和大型主机的 COBOL 程序协同工作的 Windows 用户界面工具。

12.3.4.2 用于两层客户/服务器的 SDE

如今，典型的用于两层客户/服务器应用（也称为分布式数据应用）的 SDE 包括一个基于客户端的编程语言和内建的到一个或多个服务器数据库引擎的 SQL 连接。两层客户/服务器 SDE 的例子包括 Powersoft 公司的 PowerBuilder、微软公司的 Visual Studio（客户/服务器版）和 Borland 公司的 Delphi（客户/服务器版）。通常，这些 SDE 提供了以下功能：

- 用于快速地构建图形用户界面的快速应用开发（RAD）环境，图形用户界面将被复制到所有的客户端 PC 上，并在 PC 上执行。
- 为以上的 GUI 和使用 GUI 的相关系统事件（例如鼠标单击、按键等）自动生成模板代码，程序员则只需要添加用于业务逻辑的代码。
- 编程语言将被编译复制到客户端 PC 上，并在 PC 上执行。

- 到各种关系数据库引擎的连接（以上语言）以及同那些引擎的互操作。互操作通过包含 SQL 数据库命令（例如，创建、读取、修改、删除和排序记录）来实现，将把命令发送到在服务器上执行的数据库引擎。
- 在客户端使用的复杂的代码测试和调试环境。
- 帮助程序员开发、维护和运行用户数据、行为和事件的可复用测试脚本的系统测试环境，它测试编译后的程序，以确保代码修改没有引入新的或未预见的问题。
- 报告编写环境，用来简化来自远程数据库的最终用户报告创建工作。
- 用于客户端 PC 的帮助文件著作系统。

如今，这些工具中大多数都进入了打包的 SDE 中，但是独立软件工具供应商已经出现，他们制造替代的工具，这些工具经常比基本 SDE 中提供的那些工具具有更强大的功能和/或效率。要了解更多有关这些附加工具的知识，请查询 Programmers Paradise 网站，那里有大量的软件开发工具介绍。

两层客户/服务器应用中的某些过程逻辑可以以存储过程的形式卸载到数据库服务器上。在这种情况下，存储过程用 SQL 语言的一个超集编写，然后这些过程从客户端“调用”，在服务器上执行。不同的专家有的喜欢存储过程，有的不喜欢。优点是，存储过程可以更好地强制数据库表中的数据完整性，它们是可复用的且可验证的；缺点是，存储过程模糊了框架中应用处理层和数据处理层之间的界限，而许多设计人员喜欢一种更协调的称为水平分层的设计策略。水平分层要求一个应用的表现层、应用层和数据层被物理地分离。水平分层技术据称允许修改和增强每层组件，而不会影响系统中的其他层。

12.3.4.3 用于多层客户/服务器的 SDE

企业应用开发技术的前沿是在用于三层（和更多层）客户/服务器架构的 SDE 中。与两层应用不同，n 层应用必须支持超过 100 个用户，具有类似大型主机的事务响应时间和吞吐量，以及 100GB 或更大的数据库。虽然前面描述的两层 SDE 正尽力在市场上拓展，但一种不同类型的 SDE 目前主导着市场。通常，这类 SDE 必须提供两层 SDE 具有的基本功能，再加上以下功能：

- 支持客户端和服务端异构计算平台。
- 同时用于客户端和服务器的代码生成和编程。
- 特别强调通过使用软件应用框架、模板、组件和对象实现可复用性。
- 捆绑的小 Case 工具，用于同代码生成器和编辑器互操作的分析和设计。
- 帮助分析员和程序员在客户端和服务端之间分割应用组件的工具。
- 帮助开发人员部署和管理完成的应用到客户端和服务端上的工具。这通常包括安全管理工具。
- 自动调整应用到更大规模和不同平台（客户端和服务端）的能力。
- 复杂的软件版本控制和应用管理。

n 层客户/服务器 SDE 的例子有 Dynasty 公司的 Dynasty 和 IBM 公司的 VisualAge（一个产品系列）。另外，大量的独立软件工具供应商正在为这些 SDE 构建附加的和替代的工具。

12.3.4.4 用于因特网和内联网客户/服务器的 SDE

为了支持客户/服务因特网和内联网应用，快速应用开发工具正在兴起。大多数这类语言都围绕以下 4 个核心标准技术进行构建：

- HTML（超文本标记语言）——用于构造大多数因特网和内联网网页内容和超链接的语言。
- XML（可扩展标记语言）——用于通过 Web 传输数据和属性的可扩展语言。
- CGI（计算机网关接口）——用于发布图形化万维网组件、结构和链接的标准。
- Java——一种通用程序设计语言，用于创建与平台无关的程序、servlet 和可以在浏览器的 Java 虚拟机内运行的 applet。

Java 专用 SDE 的例子包括 IBM 公司的 WebSphere、Borland 公司的 Jbuilder，这些 SDE 可以创建因特网、内联网和非因特网/内联网应用。几乎所有现有的两层和 n 层 SDE 也都发展成可以支持 HTML、XML、CGI 和 Java。

12.4 建模信息系统应用架构

使用逻辑 DFD 建模过程需求已经是广泛接受的实践，但是从面向分析的逻辑 DFD 到面向设计的物理 DFD 的转变一直有点不好理解。我们希望有一种高层次的通用设计，可以用于设计系统的应用架构以及构成系统的过程。同时，又不想因陷入一种无意义的建模练习而减慢系统设计和快速应用开发的速度。简单地说，我们想要有一个蓝图来指导我们走过详细设计和构造阶段，这个蓝图将为详细说明和快速开发确定设计单元。

12.4.1 绘制物理数据流程图

绘制物理 DFD 的机制与绘制逻辑 DFD 的机制几乎完全一样，正确性规则也一样。一个可接受的设计得到：

- 可工作的系统。
- 实现了在逻辑 DFD 中说明的用户需求的系统。
- 提供了足够性能（吞吐量和响应时间）的系统。
- 包括了足够内部控制（为了消除人为的和计算机的错误，确保数据的完整性和安全性，并满足审计约束）的系统。
- 可适应不断变化的需求和改进的系统。

可以为整个系统开发一个物理 DFD，或者为目标系统开发一套物理 DFD。我们的方法学建议采用如下方式：

- 应该为网络架构开发一个物理数据流程图，这个图中的每个过程是系统中的一个实际处理器（客户端或者服务器）。每个服务器都是处理器，但是通常显示出每个客户端是不现实的，所以每类客户端（例如订单输入职员）用一个处理器表示。
- 对于上面模型中的每个处理器，应该开发一个物理数据流程图以显示将被分配到那个处理器的事件过程（见第8章），你可能需要选择复制一些事件过程到多个处理器上。例如，订单可能在每个地区的服务器和客户端上被处理。
- 最简单的事件过程以外的所有事件过程都应该被分解成设计单元，并作为一个物理数据流程图建模。设计单元是一个自包含的过程、数据存储和数据流集合，它们共享了类似的设计属性。设计单元作为整个系统的一个子集，其输入、输出、文件和数据库以及程序可以作为一个子系统进行设计、构造和单元测试。

设计单元的例子是被设计成一个程序的一组过程（一个或多个）。然后可以把设计单元分配给单个程序员（或团队），这样他（他们）可以独立工作，而不影响其他程序员。实现的设计单元最后被装配到最终的应用系统中。可以给设计单元排列优先次序，以实现系统的不同版本。

12.4.2 网络架构

要绘制的第一个物理 DFD 是网络拓扑 DFD。网络架构 DFD 是一个物理数据流程图，它将处理器（客户端和服务端）和设备（例如机器和机器人）分配到网络中，并确定：1）客户端和服务端之间的连接；2）用户将在哪里与处理器（通常只有客户端）交互。

为了确定处理器及其位置，开发人员需要使用以下两个资源：

- 如果存在企业信息技术架构，架构可能规定了应该采用的客户/服务器模式。
- 应该征求有经验的网络管理人员和/或专家的建议，以决定什么是适当的、什么是可能的以及系统可能对计算机网络有什么影响。

网络架构 DFD（见图 12-27）需要显示某些不同于通常 DFD 的信息：它们不显示专门的每秒数据流，相反，它们显示数据流可在其上任意流动的“高速公路”。而且，网络拓扑 DFD 还说明了以下内容：

- 服务器及其物理位置。服务器并不总是放置在位置连接图中指示的地方，网络职员对服务器的访问通常是一个要考虑的因素。有些网络管理任务可以远程地完成，而有些任务还需要直接访问服务器。

- 在不同的服务器上存储特定表。在这种情况下，为了清楚起见，应该把每个表记录成物理 DFD 中的一个数据存储，并把每个表连接到相应服务器上。
- 在不同的服务器上存储特定表的子集。在这种情况下，完全同上面一样地记录表，但需要指出哪个表是整个记录集的子集。例如，标记“DB2: ORDERS TABLE (REG SUBSET)”将指示存储在 DB2 数据库表中的某个地区的所有订单的一个子集。
- 在不同的服务器上复制（重复）特定表或子集。在这种情况下，被复制的数据存储在物理 DFD 中显示。被复制的表的某个拷贝被指定为“主拷贝”，而所有其他拷贝都被指定为“拷贝”或“复件”。

为什么要分布数据存储？存在许多原因。首先，有些数据实例仅仅在局部有意义；其次，性能经常可以通过划分数据子集到多个地点而得到提高；最后，有些数据需要被本地化以确定数据的管理关系。音阶公司案例研究的数据分布和技术确定在图 12-28 中显示。

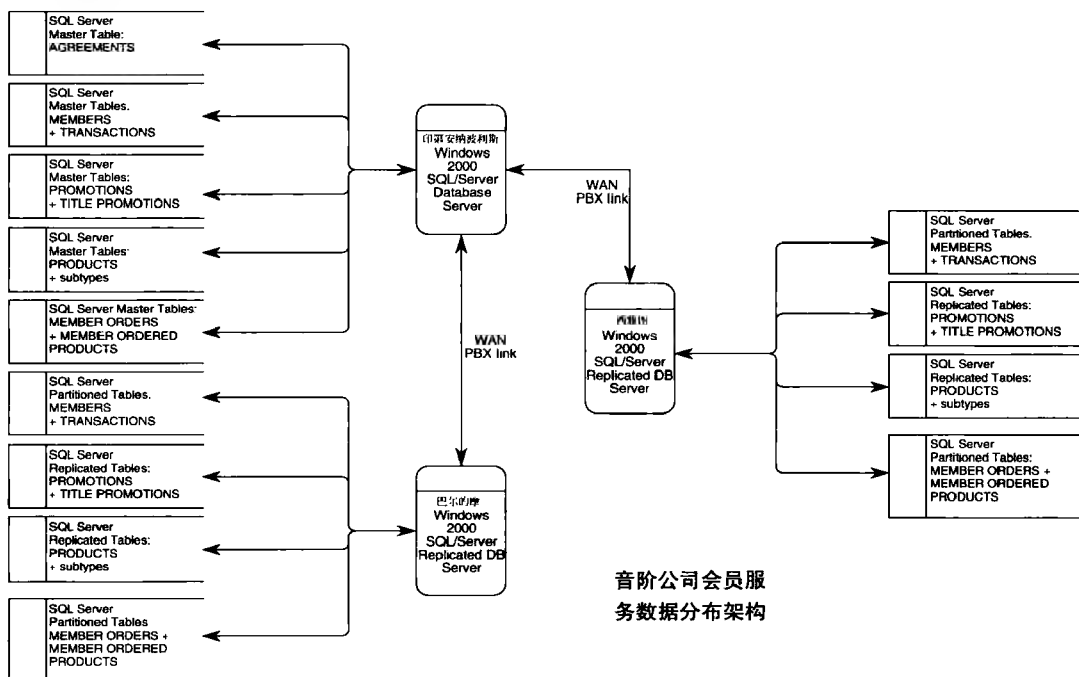


图 12-28 音阶公司案例研究的数据分布和技术确定

数据分布决策可能会很复杂——通常决策由数据和数据库专家指导，数据管理课程和课本中也会讲授。在本书中，仅考虑如何记录有关分割和复制的决策。

12.4.4 过程分布和技术确定

现在可以分配信息系统过程到处理器了，具体如下：

- 对于两层客户/服务器系统，所有逻辑事件图（见第 5 章）都被分配到客户端。
- 对于三层客户/服务器和网络计算系统，必须详细地检查每个事件的基本（详细的）数据流图，决定哪个基本过程应该被分配到客户端，哪个应该被分配到应用服务器上。通常，数据收集和编辑分配到客户端，而其他业务逻辑分配到服务器上。如果把一个逻辑 DFD 的不同部分分割到不同的客户端和服务端，应该对每个客户端和服务端上的相应部分绘制独立的物理 DFD。

分割之后，每个物理 DFD 应该对应一个给定业务事件的设计单元（业务事件或者用例见第 6 章中讨论）。对于每个设计单元，必须指定一个实现方法和用来实现过程的 SDE。另外，还必须指定数据流的实现方法。

音阶公司的会员服务系统将使用多层客户/服务器和网络计算架构实现。分配到一个客户端的一个事件的例子 DFD 如图 12-29 所示。注意图中数据存储也显示出来了, 尽管我们知道它们已经被分割到一个数据库服务器上, 但对于必须实现 DFD 的程序员是有好处的。

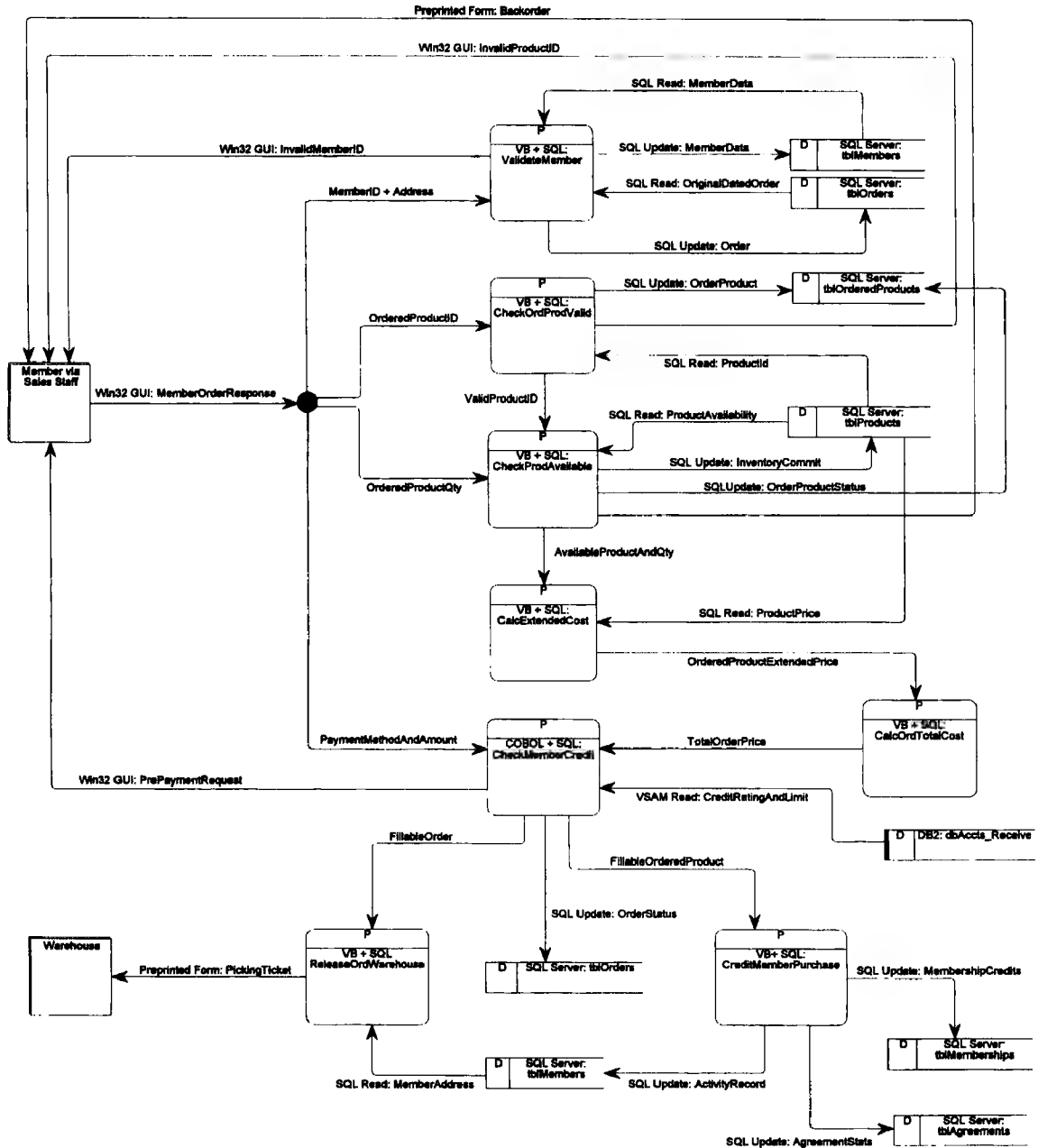


图 12-29 一个事件的物理 DFD

12.4.5 人/机边界

过程设计的最后一步是分离出物理 DFD 中表示了手工处理（而非计算机化的过程）的部分，这有时被称为“建立人/机边界”。建立人/机边界并不困难，但也不像最初想象的那样简单。困难产生在当人/机边界穿过一个逻辑过程时——换句话说，过程的一部分是手工的，而另一部分是由计算机处理

的。这种情况在逻辑 DFD 中很常见，因为绘制它们时没有考虑实现问题。

图 12-30 增加了人/机边界到一个物理 DFD，注意边界穿过了几个过程，包括 CHECK MEMBER CREDIT（检查会员信用）过程。确定边界的过程需要以下两步：

1. 手工过程部分作为一个独立设计单元（见图 12-31）。所有这些过程完全是手工的，手工设计单元到计算机处理过程（见图 12-30）的接口被描绘成外部代理。最终，设计单元中的手工过程必须清楚地向那些将执行它们的人描述出来。
2. 如果需要，原始图中的相应过程应该重新命名，以便只反映计算机处理的过程（实际上，过程已经这样命名了）。

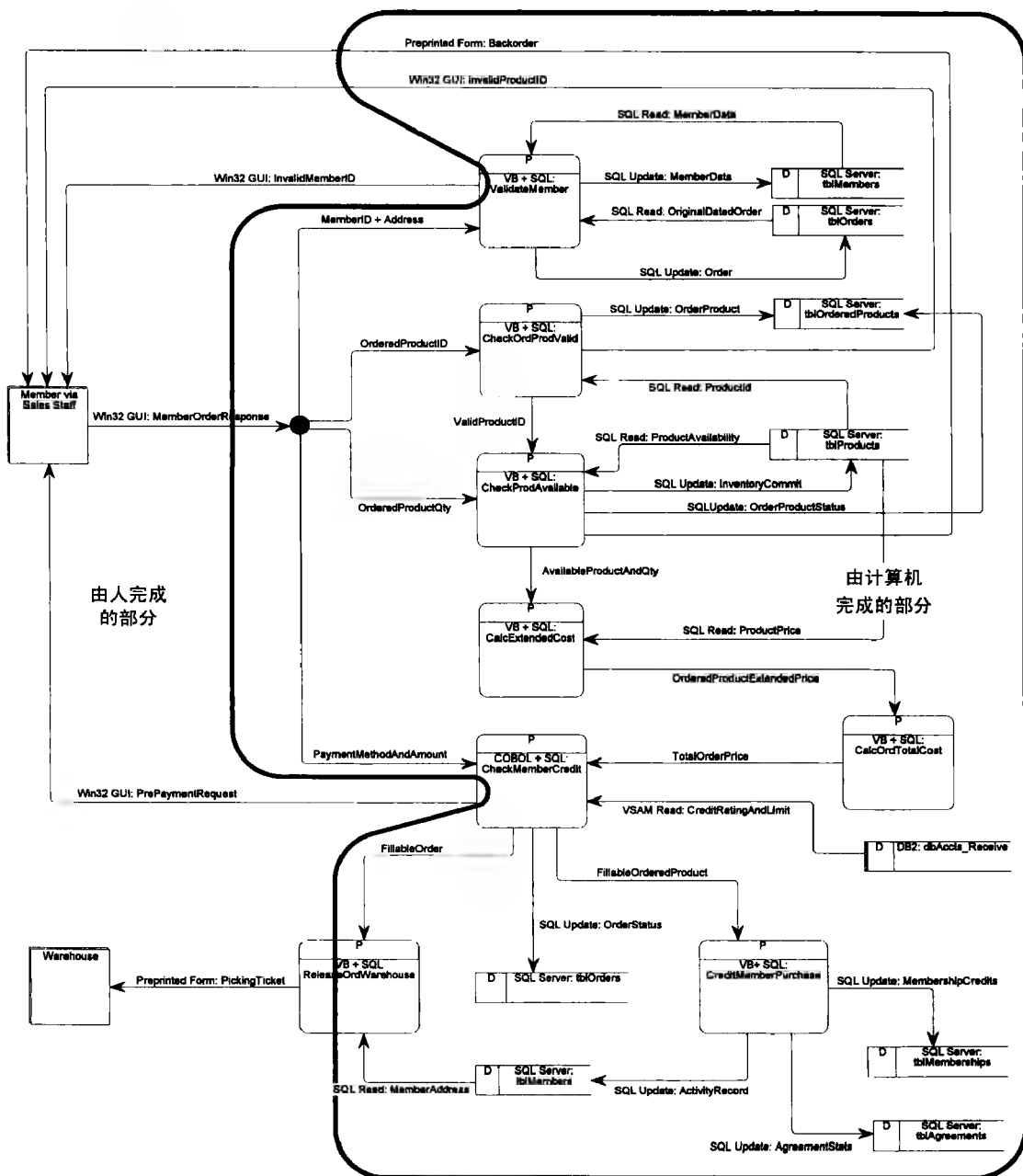


图 12-30 人/机边界

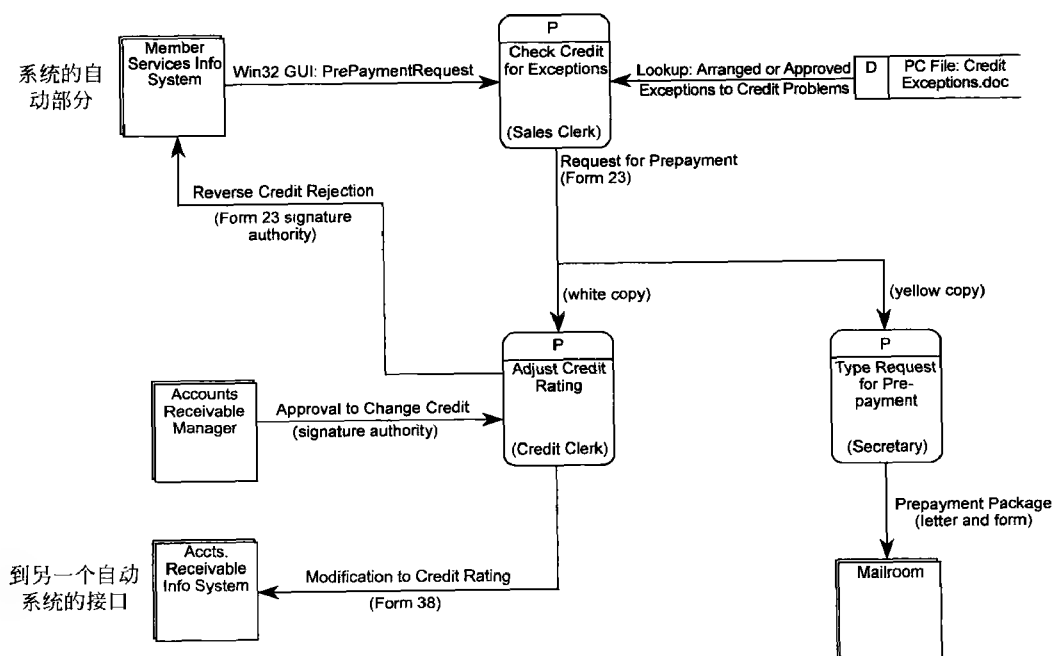


图 12-31 一个手工设计单元

复习题

1. 在传统的结构化分析和设计中，开发什么系统模型，用于什么目的？
2. 为什么完全的结构化分析和设计方法很少再被使用？
3. 当一个逻辑过程被分解成多个物理过程时，或者若干个物理过程被增加时，对于设计人员来说什么很重要，并需要检查？
4. 为什么在一个物理 DFD 中显示的物理过程数量一般都大于逻辑过程数量？
5. 物理数据流图表示了什么？
6. 设计人员在系统设计中经常忽略什么类型的数据存储？
7. 尽管集中式系统不太复杂而且更容易实现，然而分布式系统相比集中式系统被更多地采用。为什么？
8. 表示层和表示逻辑层之间有什么区别？
9. 什么是文件服务器系统，它使用什么样的网络环境？
10. 文件服务器固有的限制和缺点是什么？
11. 瘦客户端和胖客户端之间有什么区别？
12. 电子商务中使用什么网络架构？请解释每一层如何相关联。
13. 什么是建模一个信息系统的架构的高层任务序列？

问题和练习

1. 你正在一个开发公司内联网的项目的系统设计阶段的中期，项目团队正在举行一个计划会议。其中一位项目的系统用户在会上说话很少，最后他发言说：“你们这些所有技术人员一直在谈论我们将要设计的应用架构。我不清楚你们正在谈论什么。”定义和解释对于小组中的非技术的系统用户来说应用架构是什么。
2. 物理数据流图有什么用途？通常，它们同逻辑数据流图有什么区别？物理 DFD 使用哪些基本图形和连线？物理 DFD 是一种遗留的设计工具，还是在如今面向对象世界中仍可用的工具？
3. 你正在进行一个项目，该项目为一个汽车分销商设计一个新的订单系统。你正在开发物理 DFD，其中一个逻辑过程是“检查库存”。如果这个过程既可以由人执行，也可以由计算机执行，那么作为一个物理过程你将如何表示它？（注意，使用图 12-1 显示的图形格式。）如果这个过程完全由计算机执行，但使用不同的技术，那么如何处理呢？
4. 解释数据复制及其用途。你会在哪类数据库系统找到数据复制？

5. 文件服务器系统和客户/服务器系统有什么共同点？有什么差别？客户/服务器方案最重要的优点是什么？
6. 你正在一个快速增长企业的 IT 部门工作，该企业正在规划实现一个新的客户/服务器系统。最初，只有很少的不超过 100 个客户端，以及大量的穿过网络的数据输入和数据分析活动。业务驱动力将能够引入数据并很快地使系统“崩溃”。项目的预算很充足，允许购买强大的工作站和个人计算机。团队中的设计人员十分平均地分别倾向于两层和三层客户/服务器结构。他们找你提供建议。你将提出什么建议？为什么？
7. 因特网技术在过去的十几年里呈爆炸性的速度发展。在许多观点中，网络计算架构代表了一种重要的从客户/服务器架构向十分不同的方向的变化。为什么？
8. 匹配第 1 列中的术语和第 2 列中的定义或例子。
9. 批处理从 20 世纪 50 年代就开始使用，许多人认为批处理是一种过时的数据处理方法。但如果批处理是过时的，那么为什么仍有新的批处理应用被开发？
10. 匹配第 1 列中的术语和第 2 列中的定义或例子。

1. 瘦客户端	A. 病人治疗记录
2. 逻辑数据存储	B. 数据输入屏幕
3. 组件服务器	C. SQL 插入：新账号
4. 处理器	D. 数据终端
5. 事务服务器	E. SAS 文件：等待清单报告
6. 大型主机	F. CORBA
7. 表示层	G. 报告格式化应用
8. 物理数据流	H. 分布式系统
9. 物理数据存储	I. 集中式系统
10. 表示逻辑层	J. Microsoft Exchange
11. 广域网 (WAN)	K. Tuxedo
12. 对象共享标准	L. 客户
13. 引用逻辑层	M. 统计分析应用

1. 两层客户/服务器 SDE	A. HTTP
2. 设计单元	B. CICS
3. EDI	C. 物理上分离的表示层、数据层和应用层
4. 应用中间件	D. 雇员的“首页”
5. 多层客户/服务器 SDE	E. 决定应用组件的分布
6. 虚拟企业	F. Windows CE
7. 内联网客户/服务器 SDE	G. Allegris
8. 分割	H. PowerBuilder
9. 表示中间件	I. 数据流、存储和过程的自我包含集合
10. 干净分层	J. 对象请求代理
11. 笔输入	K. 联机商业银行
12. 内联网门户	L. XML
13. 事务监视器	M. Amazon. com

11. 你正在进行一个复杂的项目，在你的企业中实现一个企业级的信息系统。你快要完成从逻辑 DFD 创建物理数据流图的工作，认识到有一些手工过程和计算机过程交织在一起。是什么造成这种现象发生？你应该如何显示物理 DFD 中的手工过程，或者你需要什么来全面显示它们？
12. 给你一套一个新系统的物理 DFD 让你检查其可接受性。当检查它们时你应该问你自己什么问题？

项目和研究

1. 尽管你的朋友取笑你，但你仍是一个收集 20 世纪 50 年代到 60 年代古典乡村音乐的收集者。你的收藏现在总共有几千张不同形式的唱片。为了有助于更好地记录这些收藏品，你决定在 Microsoft Access 中开发一个简单的库存管理系统。你想能够向系统添加新的唱片，更新已有的信息，搜索一个特定唱片或者艺术家，生成各种报表。设计这个系统，使用目前学到的技术，然后绘制一个上下文数据流图和逻辑数据流图。
2. 许多企业已经实现了内联网。联系或访问几个本地的公共领域或私人领域的拥有内联网的企业。找到负责该企业的内联网的单位或者个人，并讨论它的应用架构、特征、政策、问题，等等。
 - a. 描述你联系的每个企业。
 - b. 描述每个内联网，以及雇员如何使用内联网。
 - c. 它们主要是信息型内联网，还是被用作门户——他们的桌面应用和工作中使用的任何信息系统应用——都从内联网浏览器运行吗？
 - d. 谁“拥有”企业的内联网，谁负责粘贴内容或者保持更新内联网？
 - e. 你的讨论是否使你觉得企业的雇员和内联网所有者理解他们的内联网的潜力，并正使用它发挥到全部的潜力？解释你的回答。
3. 你是一个顾问，被一个公司雇用帮助更新公司的 IT 架构规划。公司生产发电机，在整个北美和中美洲都有销售点和服务机构。你的任务之一是给公司推荐一种数据分布策略。尽管你熟悉应用架构的原理和描述架构的方法，但你还是花了些时间才准备好建议一个数据分布方法。为了做出一个关于公司应该采取的数据分布方法的恰当建

议，在网上或者学校图书馆调研你应该询问的问题和使用的的评价准则。

4. 大型主机曾经主导着信息技术，但现在已经衰退到了类似客户/服务器系统之类的其他技术的阴影之下了。似乎经常有一大堆文章报告大型主机的最终死亡。但也经常可以看到另一阵关于大型主机重生的报告。研究你学校的图书馆内、因特网上或者同其他一些有经验的 IT 经理探讨关于大型主机计算的趋势的主题。
 - a. 描述你的研究资源和他们的位置。
 - b. 你能够找到“确凿”的信息吗，例如大型主机每年的销售量？
 - c. 在公共领域和私人领域之间你发现在大型主机使用方面有什么差别吗？描述一下。
 - d. 基于你发现的信息，关于大型主机如今的状况你将得出什么结论？
 - e. 今后 10 年将如何呢？如果存在，你认为大型主机计算在公共和私人领域将扮演什么角色？论证你的观点。
5. 新的开发工具使得客户/服务器因特网应用似乎每天都在不断出现（有点夸张）。研究一个新开

发工具，以及工具使用的核心技术，例如 .NET 或 XML。然后，为你的首席信息官（真实的或者假设的）准备一份分析报告，评估这个新工具和/或技术以及在你们企业使用的潜力。注意：因为这个分析报告的目标观众是一位主管，所以你的报告应该在合适的高度涉及要点，虽然也可以包含详细技术文档的参考资料或链接。

6. 访问或者研究你所在地区的一个大型公司或政府机构，了解它的应用架构。看看你是否能够得到一份它的 IT 架构规划或一份等价文档的拷贝。
 - a. 描述你研究的企业。
 - b. 描述它的应用架构。它主要的应用架构是什么？
 - c. 它使用哪种类型的基于因特网的计算架构？它扮演多么重要的角色？
 - d. 该企业仍使用大型主机或小型机技术吗？如果使用，描述当前是如何使用的。
 - e. 在今后 5 年内，应用架构技术会有什么可以预见的变化发生吗？企业应对这些技术变化的策略是什么？你认为它的战略将有效吗？
 - f. 绘制一份它的整体信息系统架构的高层上下文图。

小型案例

1. 考虑你以前为 Wow Munchies（杂货店）研究的电子商务站点。做出任何需要的假设，进行任何需要的新研究。为 Wow Munchies 创建一个网络计算架构模型。陈述你的假设，以及你所做的背景研究，用一份短报告说明。
2. 你将发现，作为一名系统分析员，设计系统的人经常不是开发程序的人。在前面，本书指出一个可接受的物理数据流图设计导致：
 - a. 一个可工作的系统。
 - b. 满足用户需求的系统。
 - c. 提供了足够性能的系统。
 - d. 包括了足够的内部控制的系统。
 - e. 适应不断变化的需求的系统。

寻找一个看上去可接受的系统 DFD 设计的例

子，但该设计没有导出满足所陈述的需求的系统。这个问题是一个设计问题还是技术问题？程序员理解了分析员所要求的东西吗？在课堂上讨论。

3. 你是一个团队成员，该团队被指派给 VideoStore（一个电影租赁公司），设计一个系统。这个公司只在商店中出租电影（不在网上），在俄亥俄州有大约 10 个商店。逐步讨论，你将如何通过生命周期过程（包括设计阶段），来创建这个系统。尽你的可能详细说明，并使用专门的例子。
4. 回到第 5 章关于 PIECES 和第 10 章关于候选系统矩阵的材料。它们各有什么优点？使用它们一起为你在前一题目中研究的视频租赁商店考虑 3 个可能的系统。如何使用多视角矩阵提供更多的系统情况的全面视图？

团队和个人练习

1. 经理和领导之间有什么区别？寻找一个你认为是真正的伟大领导的人的例子，一个你认为是真正的出色经理的例子。他们有什么特点？你认为好的领导也将会是好的经理吗？在课堂上讨论。
2. 你认为攻击两层客户/服务器系统相比开发三层客户/服务器系统是更容易，更难还是一样的困难？找出每个架构中相应的安全弱点，在课堂上

讨论。（注意：我们不是在试图创造黑客。但很难创建一个合理的系统，选择合适的架构、语言、加密标准、软件等，而没有意识到所做的选择的安全威胁和内在弱点。）

3. 圆桌讨论：你认为存在合乎道德的黑客这样的事物吗？如果存在，将是什么样子？

数据库设计

本章概述和学习目标

数据存储是绝大多数信息系统的关键组件，有些人甚至认为它是最关键的组件。本章讲授物理数据库的设计和构造，并介绍数据库设计的工具和技术，本章将介绍以下内容：

- 定义字段、记录、文件和数据库，并给出例子。
- 描述一个现代的数据架构，其中包括文件、运行数据库、数据仓库、个人数据库和工作组数据库。
- 比较系统分析员、数据管理员和数据库管理员同数据库相关时的角色。
- 描述数据库管理系统的架构。
- 描述关系数据库如何实现逻辑数据模型的实体、属性和关系。
- 将逻辑数据模型转变成物理的关系数据库模式。
- 生成 SQL 代码以创建模式中的数据库结构。

本章关键术语

文件 (file) 是相似记录的集合。

数据库 (database) 是相关文件的集合。

字段 (field) 是存储在文件或数据库中的有意义数据的最小单元。

次键 (secondary key) 是其值标识一个记录或者所有记录的一个子集的字段。

记录 (record) 是按照预定义格式安排的字段集合。

分块因子 (blocking factor) 是包含在一个读或写操作中的逻辑记录数。

表 (table) 是文件在关系数据库中的等价。

主文件 (master file) 是包含了相对稳定的记录的表。

事务文件 (transaction file) 是包含了描述业务事件的记录的表。

文档文件 (document file) 是包含了历史数据的表。

归档文件 (archival file) 是包含了已经从联机存储中删除了的主文件记录和事务文件记录的表。

表查询文件 (table look-up file) 是包含相对静态的可被共享的数据的表。

审计文件 (audit file) 是包含了修改其他文件的记录的表。

数据架构 (data architecture) 定义如何开发文件和数据库。

运行数据库 (operational database) 支持信息系统的日常运行和业务事务处理，也称为事务数据库。

数据仓库 (data warehouse) 是存储从运行数据库中提取的数据的数据库。

数据管理员 (data administrator) 是负责数据规划、数据定义、数据架构和数据管理的数据库专家。

数据库管理员 (database administrator) 是负责数据库技术，数据库设计和构造咨询，安全、备份和恢复，以及性能调试的专家。

数据库架构 (database architecture) 是指用于支持数据架构的数据库技术。

数据库管理系统 (database management system, DBMS) 是用于创建、访问、控制和管理数据库的专用软件。

数据定义语言 (data definition language, DDL) 是 DBMS 用来定义数据库或数据库视图的语言。

数据处理语言 (data manipulation Language, DML) 是用来创建、读取、修改和删除记录的 DBMS 语言。

关系数据库 (relational database) 是一种数据库，它在一系列二维表中存储数据，这些表通过外键互相“关联”。

触发器 (trigger) 是嵌入在表中的程序，当修改另一个表时，它就被自动调用。

存储过程 (stored procedures) 是嵌入在表中的程序, 可以从一个应用程序调用。

数据库模式 (database schema) 是表示数据库的技术实现的模型或蓝图。

访问完整性 (referential integrity) 确保一个表中的一个外键值匹配相关表中的主键值。

角色名称 (role name) 是外键的名字, 反映了外键在表中的用途。

13.1 系统分析员的数据库概念

在继续后面内容之前, 需要重申: 许多对数据库设计来说重要的概念和问题在数据库和数据管理课程中讲授, 绝大多数信息系统课程表中都至少包括一门这样的课程, 我们不想用本章代替那门课程。信息系统专业的学生应该主动地选修专门介绍数据管理和数据库技术的课程, 那些课程介绍的相关技术和技巧比本章中介绍的多得多。

也就是说, 我们将首先介绍系统分析员在信息系统设计中必须掌握的数据库概念和问题。尽管本章的重点是数据库设计, 但是有经验的读者将会发现许多概念超越了文件和数据库范畴。

所有的信息系统都要创建、读取、修改和删除 (有时简称为 CRUD) 数据。数据存储的文件和数据库中。文件是相似记录的集合。例如: 客户文件 (CUSTOMER FILE)、订单文件 (ORDER FILE) 和产品文件 (PRODUCT FILE)。数据库是相关文件的集合。关键词是相关。数据库不只是文件的集合, 每个文件记录必须允许存在到其他文件记录的关系 (可以把它们考虑成“指针”)。例如, 销售数据库可能包含“链接到”对应的客户和产品记录的订单记录。

13.1.1 字段

在文件和数据库中都存在字段。字段是一个数据属性 (见第 7 章) 的物理实现。字段是存储在文件或数据库中的有意义数据的最小单元。系统可以存储 4 类字段: 主键、次键、外键以及描述性字段或非键字段。

主键是其值唯一地确定了文件中的一个记录的字段 (参见第 7 章)。例如, CUSTOMER NUMBER 唯一地标识了数据库中的一个 CUSTOMER 记录, ORDER NUMBER 唯一地标识了数据库中的一个 ORDER 记录。另外, 可以通过组合两个或多个字段创建主键 (称为复合键)。

次键是数据库的替代标识符。次键的值可以标识一个记录 (同主键一样) 或者所有记录的一个子集 (例如 ORDER STATUS [订单状态] 字段的值为 back-ordered [预订单] 的所有订单记录)。在数据库中, 一个文件可能只有一个主键, 但可以有几个次键。为了方便查询和排序, 经常为键创建索引。

外键是指向数据库中另一个文件记录的指针 (也参见第 7 章)。外键说明了数据库如何把一种类型的记录链接到另一种类型的记录。例如, ORDER RECORD 记录包含了外键 CUSTOMER NUMBER, 用来“标识”或者“指向”同 ORDER 记录相关的 CUSTOMER 记录。注意一个文件中的外键要求在另一个文件中存在对应的主键——否则, 它就不“指向”任何内容了! 因此, 为了链接 ORDER 文件和 CUSTOMER 文件, ORDERS 文件中的 CUSTOMER NUMBER 字段要求在 CUSTOMERS 文件中也存在 CUSTOMER NUMBER 字段。

描述性字段是其他 (非键的) 存储业务数据的字段。例如, 给定 EMPLOYEES 文件, 一些描述性字段包括 EMPLOYEE NAME、DATE HIRED、PAY RATE 和 YEAR-TO-DATE WAGES。当在系统分析中进行数据建模时, 就需要定义对键和描述符的业务需求 (见第 7 章)。

13.1.2 记录

字段被组织成记录, 文件和数据库中都存在记录。记录是按照预定义格式安排的字段集合。例如, CUSTOMER RECORD 可以由以下字段描述:

CUSTOMER (NUMBER, LAST-NAME, FIRST-NAME, MIDDLE-INITIAL, POST-OFFICE-BOX-NUMBER, STREET-ADDRESS, CITY, STATE, COUNTRY, POSTAL-CODE, DATE-CREATED, DATE-OF-LAST-ORDER, CREDIT-RATING, CREDIT-LIMIT, BALANCE, BALANCE-PAST-DUE...)

在系统设计期间，记录将被分成固定长度记录和可变长度记录。绝大多数数据库技术强制使用固定长度记录结构，这意味着每个记录实例都具有相同的字段、相同数量的字段和相同的逻辑长度，但是有些数据库会压缩未使用的字段和值，以节省磁盘存储空间。在数据库设计中，数据库设计人员一般必须理解和说明这种压缩。

在以前学过的程序设计课程中（特别是 COBOL），你可能已经遇到过可变长度记录结构，这种结构允许同一个文件的不同记录具有不同长度。例如，可变长度订单记录可能包含某些每个记录出现一次的公共字段（例如，ORDER NUMBER、ORDER DATE 和 CUSTOMER NUMBER），以及另一些根据订单中销售的产品数量而重复不同次数的字段（例如，PRODUCT NUMBER 和 QUANTITY ORDERED）。数据库技术一般不允许（或者至少不鼓励）使用可变长度记录，这种限制并不会造成设计困难，对此我们将在本章后面讲解。

当计算机程序从数据库中“读”记录时，它实际上一次访问了一组或一块（或一页）记录，这种方式最小化了实际的磁盘访问次数。分块因子是包含在一个读或写操作中的逻辑记录数（从计算机的角度）。块有时称为物理记录。如今，分块因子通常由所选择的数据库技术决定和优化，但资深数据库管理员可以调整分块因子的性能。数据库性能调试的内容最好在—门数据库课程或课本中讲授。

13.1.3 文件和表

相似的记录被组织成文件。在数据库系统中，文件经常被称为表。文件是给定记录结构的所有具体值集合。表是文件在关系数据库中的等价。常规文件和表包括以下类型：

- **主文件**或表包含了相对稳定的记录。因此，一旦将记录添加到主文件中，它就无限期地保留在系统中。记录字段的值将在其生命期中变化，但单个记录无限期地保留。主文件和表的例子有：客户、产品和供应商（SUPPLIERS）。
- **事务文件**或表包含了描述业务事件的记录，描述这些事件的数据通常具有有限的有效生命期。例如，INVOICE（发票）记录一般在发票被支付或者作为无效发票被取消之前是有用的。在信息系统中，事务记录经常保持联机一段时间，在其有效生命期之后，它们被脱机地归档。事务文件的例子有：订单、发票、申请（REQUISITIONS）和注册（REGISTRATIONS）。
- **文档文件**和表存储了历史数据，这是为了便于访问和检查，而又不必重新生成文档。
- **归档文件**和表包含了已经从联机存储中删除了的主文件和事务文件记录。所以，很少删除记录；它们只是从联机存储移到了脱机存储中。归档需求来自政府法规和后续的审计或分析需要。
- **表查询文件**包含相对静态的数据，它可以被应用程序共享以维护其一致性并改进性能，例如：销售税表（SALES TAX TABLES）、邮政编码表（ZIP CODE TABLES）和收入税表（INCOME TAX TABLES）。
- **审计文件**是修改其他文件的特殊记录，特别是修改主文件和事务文件的记录，它们与归档文件一起用于恢复“丢失”的数据。审计跟踪一般在较高级的数据库技术中构建。

不久以前，文件设计方法要求分析员精确地说明数据库中的记录应该如何排序（称为文件组织）和访问（称为文件访问）。在如今的数据库环境中，数据库技术自身通常预先定义和/或限制了包含在数据库中的所有表的文件组织。接受过培训的数据库管理员可以拥有一定的控制权限，对文件组织结构、存储位置和访问方法进行性能调整。

13.1.4 数据库

如前所述，独立的应用专用文件曾经是大多数信息系统的活力源泉，但是的确它们正慢慢地被数据库所代替。可以简单地把数据库看作是一组相互关联的文件，所谓相互关联，是指一个文件中的记录可以关联或者链接另一个文件中的记录。

例如，学生记录可以链接到那个学生的所有课程记录；反过来，课程记录可以链接到指示了上那门课程的学生记录。这种双向链接及其灵活性使得我们没必要在不同记录类型中冗余地存储相同数据库字段。因此，将多个文件合并成一个文件——数据库。

不同数据集合之间的关系这个概念在第7章中介绍了。在那一章中，学会了获取系统的数据需求，

并用实体和关系建模需求，数据库现在则提供了那些实体和关系的技术实现。

现在，如此多的应用正围绕着数据库技术构建，以至于数据库设计能力成为了分析员的一项重要技能。信息系统的历史已经得出了一个必然结论：

数据是必须被控制和管理的资源！

13.1.4.1 数据架构

数据成为数据库环境中的企业资源，信息系统围绕这个资源构建，为计算机程序员和最终用户提供了灵活的数据访问。企业的**数据架构**定义了企业将如何开发和使用文件和数据库以存储组织中的所有数据；要使用的文件和数据库技术；以及管理数据资源的管理机构。

图 13-1 演示了数据架构，许多公司的数据架构就是从中发展来的。如图 13-1 所示，大多数公司仍将有大量基于常规文件的信息系统应用，其中大部分是在高性能数据库技术出现之前开发的。在许多情况下，这些文件的处理效率较高，或者重新设计这些文件的预计费用太高，所以它们还没有被转换成数据库。

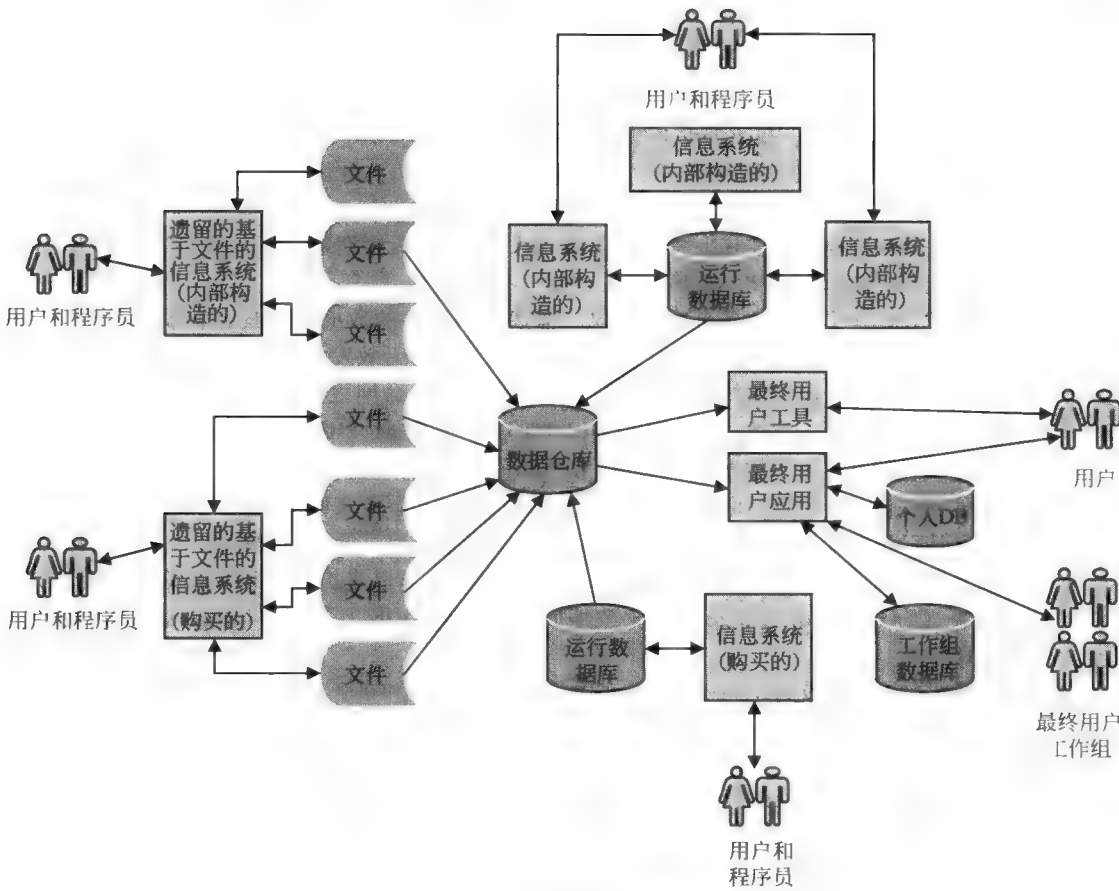


图 13-1 典型的现代数据架构

如图 13-1 所示，企业开发运行（或事务）数据库，以支持主要信息系统的日常运行和业务事务处理，这些系统被开发（或购买），以代替以前支持应用的常规文件。当这些数据库的访问被限制在计算机程序时，它们使用 DBMS 来处理事务、维护数据并生成日常调度的管理报告，并且提供一些查询访问功能。

许多信息系统对是否给最终用户提供到运行数据库的访问，供查询和报告之用十分犹豫，因为计划外的报告和查询可能会使计算机负载过重，并超过数据库设计支持的业务操作能力。因此，人们可

以在独立的计算机上开发数据仓库。

数据仓库存储从运行数据库和常规文件中提取的数据。然后,用户可以使用第4代编程语言、查询工具和决策支持工具从这些数据仓库中产生报告和分析,这有时被称为数据挖掘。

图13-1也显示了个人和工作组(或部门)数据库。个人计算机和局域网数据库技术已经快速地成熟,这使得最终用户可以开发个人数据库和部门数据库。这些数据库可以包含独特的数据,或者可以从常规文件、运行数据库和/或数据仓库导入数据。个人数据库使用PC数据库技术(例如Access、dBASE和Visual FoxPro)构建。

现代数据架构还包含支持因特网的数据库技术,例如Oracle 10g为支持Web的数据库提供了特殊工具。

无可否认,这个结构是先进的,而且许多公司正在使用这个结构的变种。为了管理企业级数据资源,数据库专家队伍可以围绕以下管理员组织起来:**数据管理员**负责数据规划、数据定义、数据架构和数据管理。一个或多个**数据库管理员**负责数据库技术,数据库设计和构造咨询,安全、备份和恢复,以及性能调试。在小型企业中,这些角色可以被组合或分配给一个或多个系统分析员。

13.1.4.2 数据库架构

到目前为止,我们已经提到了几种数据库技术,它们使得前面介绍的数据架构成为可能。**数据库架构**是指数据库技术,包括数据库引擎、数据库工具、用于分析和设计的数据库CASE工具以及数据库应用开发工具。数据库架构的控制中心是其数据库管理系统。

数据库管理系统是从计算机供应商那里得到的专门的计算机软件,它用于创建、访问、控制和管理数据库。DBMS的核心经常被称为数据库引擎,引擎响应专门的命令以创建数据库结构,然后创建、读取、修改和删除数据库中的记录。数据库管理系统从数据库技术供应商那里购买,这些供应商包括Oracle公司、IBM公司、微软公司或Sybase公司。

图13-2描述了一个典型的数据库管理系统架构。系统分析员或者数据库分析员按照记录类型、这些记录类型中包含的字段和记录类型之间存在的关系设计数据结构,这些数据结构则使用数据定义语言定义到数据库管理系统中。DBMS使用**数据定义语言**(DDL)创建记录类型、字段和结构化关系。另外,DDL还定义了数据库视图,视图限制了数据库可以被不同用户和程序使用或访问的部分。

绝大多数数据库管理系统都存储用户数据和元数据——关于数据的数据(或说明)——例如,记录和字段定义、同义词、数据关系、验证规则、帮助消息等。有些元数据存储在实际的数据库中,而另一些元数据存储 CASE 工具资料库中。

为了有助于设计数据库,CASE工具可以由数据库技术供应商提供(例如Oracle公司的Designer),或者由第三方CASE工具供应商提供(例如,Popkin公司的System Architect、微软公司的Visio Enterprise或者Computer Associates公司的ERwin等)。

数据库管理系统还提供了一种数据处理语言,用于访问和使用应用中存储的数据。**数据处理语言**(DML)用来创建、读取、修改和删除数据库中的记录,并用来在不同记录和记录类型之间导航——例如,从客户记录到这个客户的订单记录。DBMS和DML隐藏了关于记录如何组织和分配到磁盘的细节。DML一般很灵活,它自身可以用来创建、读取、修改和删除记录,或者其命令可以从一个独立的主机使用编程语言“调用”,这些编程语言包括COBOL、Visual Basic或Java。

许多DBMS不要求使用DDL构造数据库,也不要求使用DML访问数据库。相反,它们提供专用的工具和命令执行这些任务,基于PC的DBMS(例如Microsoft Access)尤其如此。Access提供了一个简单的图形用户界面用于创建表,并提供了一个基于表单的环境和脚本语言(Visual Basic for Applications)用于访问、浏览和维护表。

许多DBMS还包括专用的报告编写和查询工具,通过这些工具用户可以不直接使用DML就能访问和格式化数据。许多高端DBMS被设计成与常用的第三方事务处理监视器(例如IBM公司的CICS和微软公司的Transaction Server)交互。

所有上面这些技术都在图13-2中显示。如今,几乎所有开发的新数据库都在使用关系数据库技术。

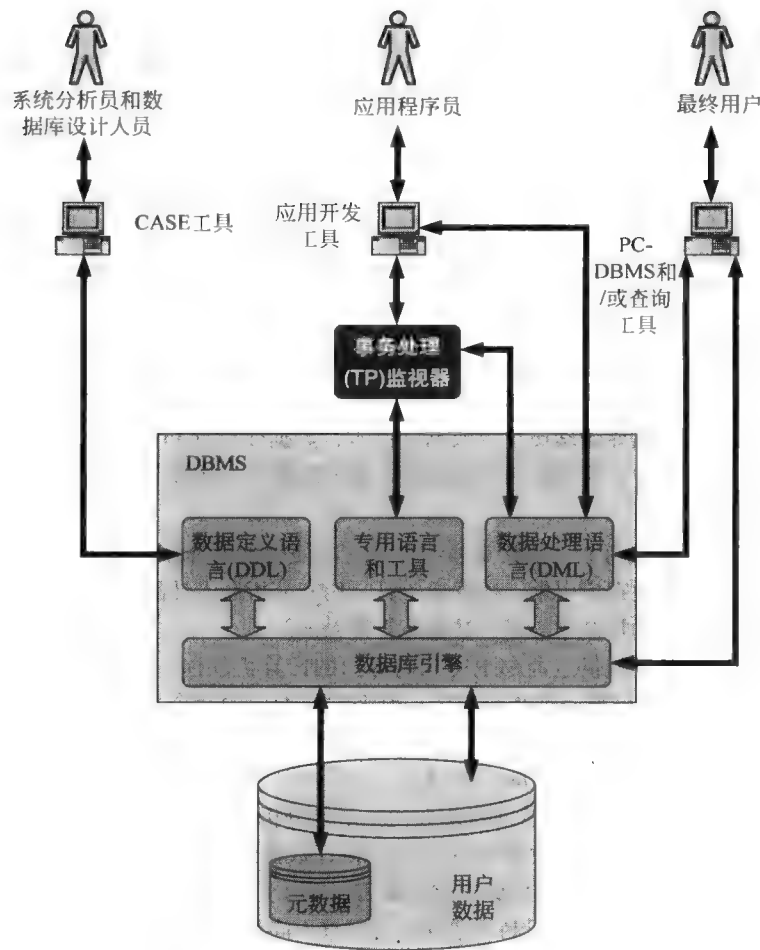


图 13-2 典型的数据库管理系统架构

13.1.4.3 关系数据库管理系统

数据库管理系统有多种类型，可以按照其结构化记录的方式进行分类。早期的数据库管理系统按照用索引和链接列表实现的层次或网络组织记录，你可以在数据库课程中进一步学习这些内容。但如今，绝大多数成功的数据库管理系统都是基于关系技术的。关系数据库在一系列二维表中存储数据，这些表通过外键互相“关联”。每个表（有时称为关系）由命名列（它们是字段或属性）和任意数量的未命名行（它们对应记录）构成。

图 13-3 演示了一个逻辑数据模型，图 13-4 是实现了这个数据模型的物理关系数据库（称为模式）。在关系数据库中，将文件看作是简单的二维表，也称为关系，其中行是记录，列对应字段。

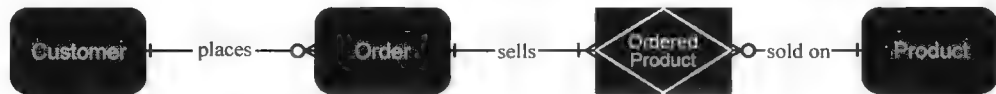


图 13-3 一个简单的逻辑数据模型

在有关系统设计和数据库的书中通常会遇到以下的表速记符号：

```
CUSTOMERS ( CUSTOMER-NUMBER, CUSTOMER-NAME, CUSTOMER-BALANCE, ... )
ORDERS ( ORDER-NUMBER, CUSTOMER-NUMBER (FK), ... )
```

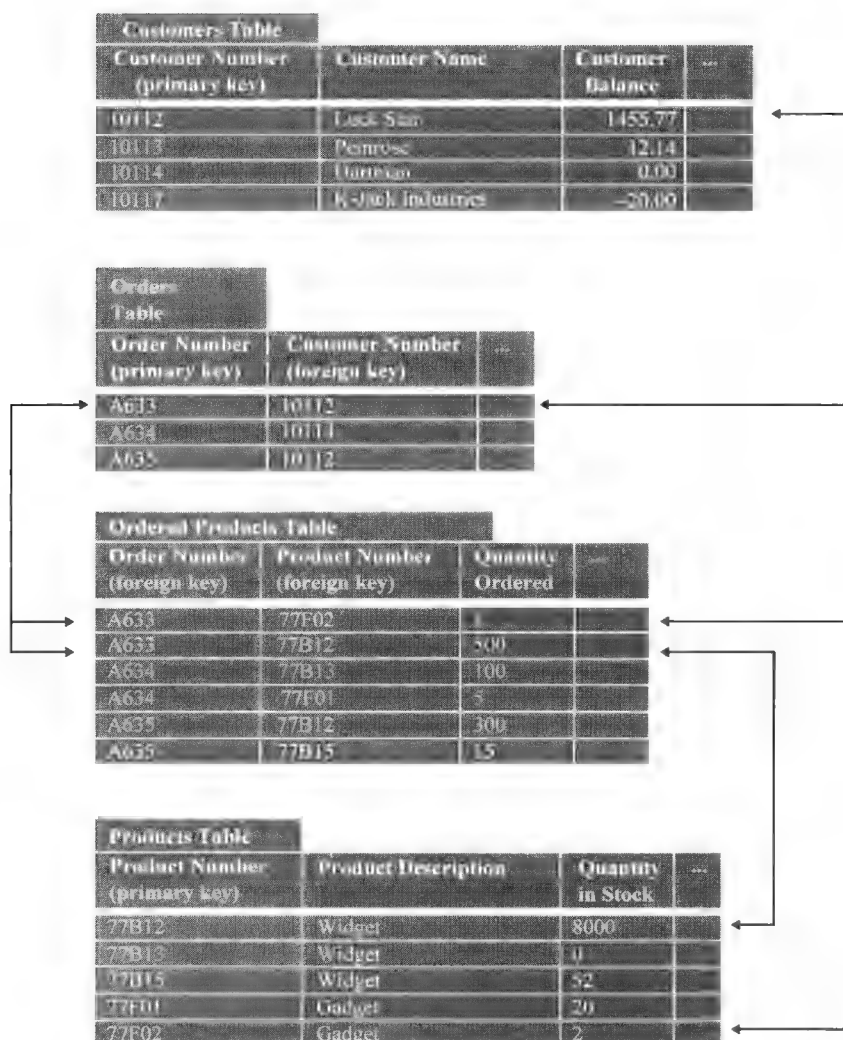


图 13-4 一个简单的物理数据库模式

ORDERED-PRODUCTS (ORDER-NUMBER (FK), PRODUCT-NUMBER (FK), QUANTITY-ORDERED, ...)

PRODUCTS (PRODUCT-NUMBER, PRODUCT-DESCRIPTION, QUANTITY-IN-STOCK, ...)

大多数关系数据库的 DDL 和 DML 都称为 SQL (有些人读作 S-Q-L, 有些人读作 *sequel*), SQL 支持全部的数据库创建、维护和使用操作。为了访问表和记录中的数据, SQL 提供了以下基本命令:

- 从表中根据特定的准则 SELECT (选择) 特定的记录 (例如, SELECT CUSTOMER WHERE BALANCE > 500.00)。
- 从表中 PROJECT (投射) 出特定的字段 (例如, PROJECT CUSTOMER TO INCLUDE ONLY CUSTOMER-NUMBER, CUSTOMER-NAME, BALANCE)。
- 通过公共字段——主键或外键——JOIN (连接) 两个或多个表 (JOIN CUSTOMER AND ORDER USING CUSTOMER-NUMBER)。

当组合使用时, 这些基本命令可以满足绝大部分数据库需求。SQL 的基本特点是命令返回一组记录, 而不仅仅返回一个记录 (如在非关系数据库和文件技术中那样)。SQL 数据库还提供了用于创建、修改、删除和排序记录的命令。

高端关系数据库还扩展了 SQL 语言, 以支持触发器和存储过程。触发器是嵌入在表中的程序, 当修改另一个表时, 它就被自动地调用。例如, 如果从航班 (PASSENGER AIRCRAFT) 表删除一个记录, 触发器可以强制自动为航班删除座位 (SEATS) 表中的所有记录。存储过程是嵌入在表中的程序, 它可以从一个应用程序调用。例如, 数据验证算法可以嵌入到表中, 在实际存储新记录和修改的记录之前, 确保它们包含了有效数据。存储过程用 SQL 的专用扩充语法 (例如微软公司的 Transact SQL 或 Oracle 公司的 PL/SQL) 编写。

触发器和存储过程都可以复用, 因为它们和表自身 (作为元数据) 存储在一起, 这使得应用程序员没有必要在每个使用这些表的应用中创建等价的逻辑。

所有的高端关系数据库管理系统 (例如, Oracle、UDB/DB2 和 SQL Server) 和许多个人计算机关系数据库管理系统 (例如微软公司的 Access) 都支持 SQL 语言标准。

高性能关系 DBMS 的例子包括 Oracle 公司的 Oracle、IBM 公司的 DB2、微软公司的 SQL Server (被用于音阶公司项目) 和 Sybase 公司的 Sybase, 这些数据库中有许多在大型主机、小型计算机和网络数据库服务器上运行。大多数个人计算机 DBMS 也是关系型的 (或者至少部分的是), 例如微软公司的 Access 和 Visual FoxPro, 这些数据库引擎可以在独立的个人计算机上运行, 也可以在局域网文件服务器上运行。图 13-5 演示了一个关系数据库管理系统的用户界面。

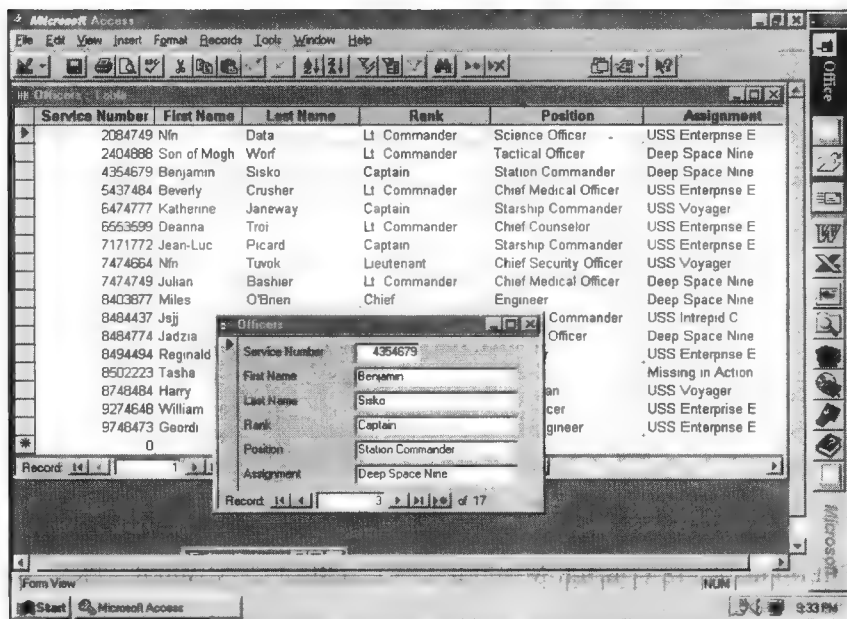


图 13-5 关系型 PC DBMS (微软公司的 Access) 的用户/设计人员界面

13.2 数据库设计的前置条件——规范化

在第 7 章中, 你学到了如何为信息系统建模数据需求, 该模型采用了一种具有完整属性的实体关系图和元数据资料库的形式。第 7 章也讲授了一种称为数据分析或规范化的技术, 这个技术用来产生满足以下质量准则的数据模型:

- 好的数据模型是简单的。作为一条通用规则, 描述一个实体的数据属性应该仅描述那个实体。
- 好的数据模型本质上是无冗余的。这意味着每个数据属性 (除了外键) 最多只描述一个实体。
- 好的数据模型应该是灵活的, 而且对于未来需求具有适应性。在没有这条准则时, 我们往往会仅仅为满足今天的业务需求设计数据库。

那么如何实现上述目标呢? 如何设计可以适应未来无法预测的需求的数据库呢? 答案就在数据分析中。

规范化是一种包括三个步骤的技术，它依次把数据模型带入第一范式、第二范式和第三范式。只有当底层的逻辑数据模型至少是 3NF 时，才能进行数据库设计。要得到更详细的解释，读者可以复习第 7 章。

13.3 现代数据库设计

任何数据库的设计通常都需要 DBA 和数据库人员的参与，由他们来处理技术细节和交叉应用问题。对于系统分析员来说，理解关系数据库的基本设计原理仍然是有用的。

这里介绍设计规则（即通常所说的指南），但不可能介绍每个细节。而且，因为音阶公司已经选择使用微软公司的 SQL Server 作为其数据库管理系统，所以我们的设计将受到这个技术的制约——每个关系型 DBMS 都表现了其性能和缺点。不过这里介绍的指南十分通用，可以应用于大多数 DBMS 环境。数据库课程和课本往往介绍更多的技术和问题。

计算机辅助系统工程（Computer-Assisted Systems Engineering, CASE）一直是贯穿本书的一个主题。市场上存在专门涉及数据库分析和设计的 CASE 产品（例如 Computer Associates 公司的 ERwin），而且绝大多数通用 CASE 工具现在都包括数据库设计工具。在这个例子中，我们继续使用 Popkin 公司的 System Architect CASE 产品，用于音阶公司案例研究。最后，绝大多数 CASE 工具（包括 System Architect）可以自动为绝大多数常用的数据库管理系统生成用来创建数据库结构的 SQL 代码，这种代码生成能力可以节省大量时间。

13.3.1 数据库设计的目标和前置条件

数据库设计的目标如下：

- 数据库应该提供对数据的有效存储、修改和访问。
- 数据库应该可靠——存储的数据应该具有高度的完整性，以促进用户信任数据。
- 数据库应该可适应和可扩展未预料到的新需求和新应用。
- 数据库应该支持信息系统的业务需求。

系统的逻辑数据模型（在案例中为一个具有完整属性的规范化的实体关系图）作为数据库设计的前置条件。来自第 7 章的这个模型在图 13-6 中重新绘制，这个模型中的每个属性必须定义数据类型、字段和默认值，这些性质同样在第 7 章中进行了介绍。

13.3.2 数据库模式

数据库的设计被描述成数据库模式的特殊模型。数据库模式是数据库的物理模型或蓝图，它代表了逻辑数据模型的技术实现（System Architect 称之为物理数据模型）。

注意 这里应该提醒注意一些容易混淆的词汇。这里以与本书前面一致的方式使用词汇“逻辑”和“物理”，但是许多数据库书籍使用词汇“概念”（我们所说的逻辑）和“逻辑”（我们所说的物理）。

关系数据库模式按照表、键、索引和完整性规则定义数据库结构，它根据所选数据库管理系统的能力、词汇和约束说明各项细节，每个 DBMS 都支持不同的类型、完整性规则等。

逻辑数据模型到物理关系数据库模式的转换有一些相当通用的规则，这些规则和指南总结如下：

1. 每个基本实体、关联实体和弱实体都被实现成一个独立表，表名可能需要按照 DBMS 的命名规则和大小限制进行格式化。例如，命名为 MEMBER ORDERED PRODUCT 的逻辑实体可能被变换成名为 tblMemberOrdProd 的物理表。前缀和空间压缩与现代程序设计语言中采用的命名标准和指南一致。
 - a. 标识主键，并且实现成表中的一个索引。
 - b. 每个次键实现成表中的索引。
 - c. 对于任何被确定为子集准则需求（见第 7 章）的非键属性，应该建立一个索引。
 - d. 如此实现每个外键，这些外键实现了数据模型上的关系，并使得表可以在 SQL 和应用程序中被连接。

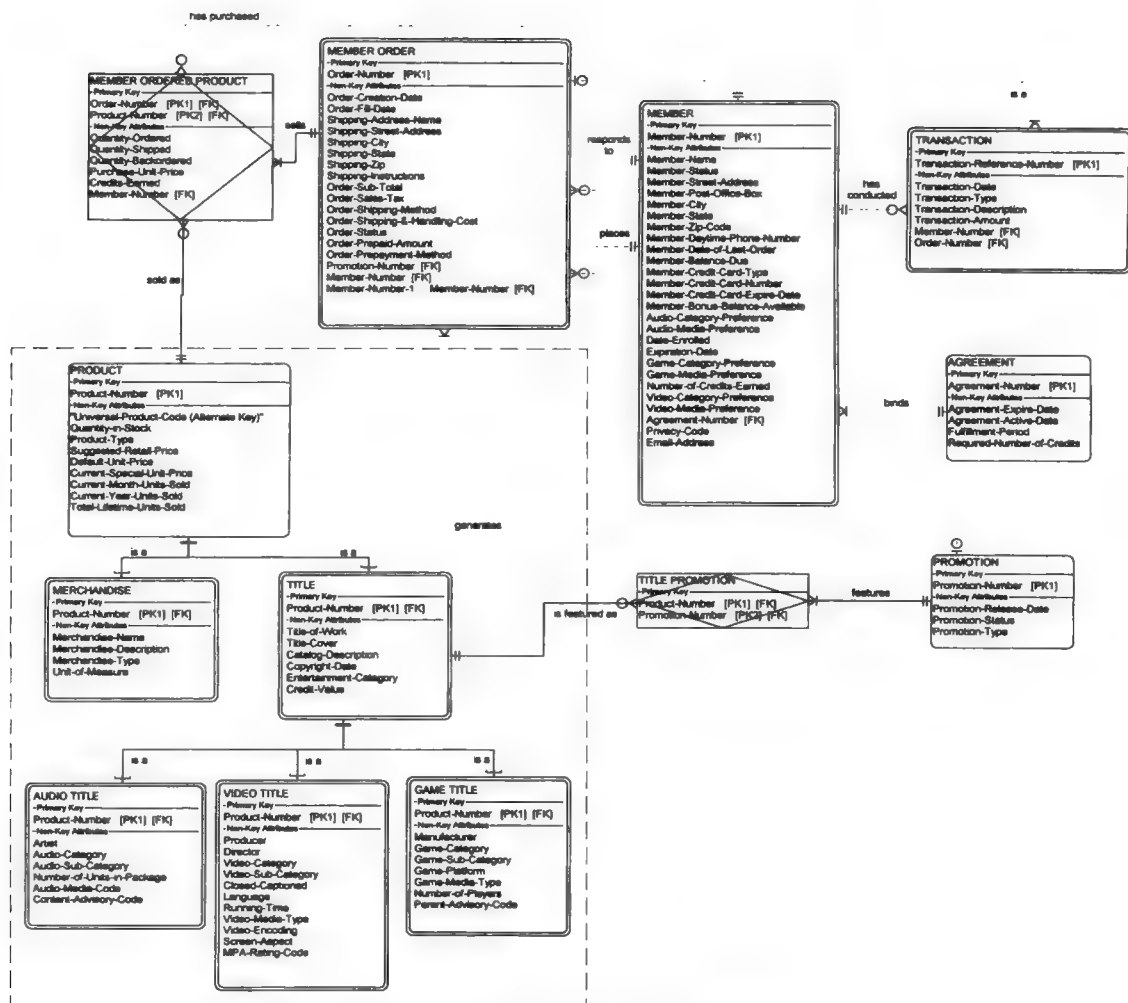


图 13-6 第三范式的音阶公司逻辑数据模型

e. 属性将用字段实现。这些字段对应了表中的列，通常必须对每个属性说明下面的技术细节（CASE 工具可以从数据模型中的逻辑描述自动地推断出这些细节）。

字段名称可能必须按照 DBMS 的约束和内部规则缩短和重新格式化，例如：在逻辑数据模型中，大部分属性以实体名称（如 MEMBER NAME）为前缀，而在物理数据库中，可能会简单地使用名字（NAME）。

- i. 数据类型。每个 DBMS 支持不同的数据类型和术语，图 13-7 显示了一些不同的数据库管理系统的的不同数据类型。
- ii. 字段的大小。不同的 DBMS 表达实数的精度不同，例如：在 SQL Server 中，NUMBER (3, 2) 的大小规定支持从 -9.99 ~ 9.99 的精度范围。
- iii. 空或非空。在记录可以提交到存储之前，字段必须有值吗？不同的 DBMS 可能使用不同的预留字表示这个特性。按照定义，主键从来不允许值为空。
- iv. 域。许多数据库管理系统可以自动编辑数据，以确保字段包含合法数据，这对于独立于应用程序确保数据完整性可能是一个很大的优点。但对于支持数据完整性的 DBMS 来说，完整性规则必须精确地在 DBMS 可以理解的语言中说明。
- v. 默认值。当用户或程序员创建了一个包含空值字段的记录时，许多数据库管理系统允许使用默认值来自动地设置值。在有些情况下，空可作为默认值。

逻辑数据类型 (存储在字段中)	物理数据类型 Microsoft Access	物理数据类型 Microsoft SQL Server	物理数据类型 Oracle
固定长度字符数据 (用于具有相对固定长度字符数据的字段)	TEXT	CHAR (size) 或 character (size)	CHAR (SIZE)
可变长度字符数据 (用于存储字符数据但大小变化很大的字段——例如“地址”)	TEXT	VARCHAR (max size) 或 Character varying (max size)	VARCHAR2 (max size)
超长字符数据 (用于大段的描述和注解, 通常每个记录不会有多个这样的字段)	MEMO	TEXT	LONG VARCHAR 或 LONG VARCHAR2
整数	NUMBER	INT (size) 或 integer 或 smallinteger 或 tinyinteger	INTEGER 或 NUMBER (size) 或 smallint 或 byte
小数	NUMBER	DECIMAL (size, decimal places) 或 NUMERIC (size, decimal places)	DECIMAL (size, decimal places) 或 NUMBER (size, decimal places) 或 NUMBER
货币	CURRENCY	MONEY 或 SMALLMONEY	见小数
日期 (含时间)	DATE/TIME	DATETIME 或 SMALL-DATETIME 取决于需要的精度	DATE
当前时间 (用来存储取自计算机系统时钟的日期和时间)	不支持	TIMESTAMP	TIMESTAMP
Yes 或 No; True 或 False	YES/NO	BIT	使用 CHAR (1) 并设置一个 yes 或 no 字段
图像	OLE OBJECT	IMAGE	LONGRAW
超链接	HYPERLINK	VARBINARY	RAW
设计人员是否可以定义新数据类型	NO	YES	YES

图 13-7 不同数据库技术的部分物理数据类型

vi. 以上说明中的各项内容作为一个完整逻辑数据模型的一部分被记录下来。如果这个数据模型是用 CASE 工具开发的, 则 CASE 工具就可以将数据模型自动地翻译成为所选数据库技术的物理语言。

2. 超类/子类实体表示其他选项, 列举如下:

- 每个超类和子类可以用一个独立表实现 (都具有同样的主键)。
- 如果子类具有类似的大小和数据内容, 则数据库管理员可以选择合并子类成为超类, 以创建一个表, 这对设置默认值和检查字段会有一些问题。在高端 DBMS 中, 这些问题可以通过为表在存储过程中嵌入默认值和字段逻辑克服。
- 超类的属性可以复制到每个子类的表中。
- 可以使用以上选项的某些组合。

3. 评价并说明访问完整性约束 (在下一节描述)。

音阶公司案例的数据库模式由 CASE 工具 System Architect 自动从逻辑数据模型中产生, 如图 13-8 所示。请注意图中下列注释:

- 每个圆角矩形定义了一个表, 矩形中命名行实际上对应要为此表创建的命名列。
- 音阶公司已经为表和列定义了一个标准命名规则, 这个规则参考了被称为匈牙利命名法的程序设计指南。每个对象的命名都不含空格、连字符和下划线, 而且每个对象都有一个前缀, 这个前缀用来定义同类对象。对于数据库对象, 使用以下标准:

tbl 指示数据库表。

col 指示表中一列。

虽然在模式中没有描述, 但其他公共数据库前缀也可以包含数据字典 (资料库) 底层的模式中来, 以便那些前缀可用于生成正确代码。可能的前缀包括:

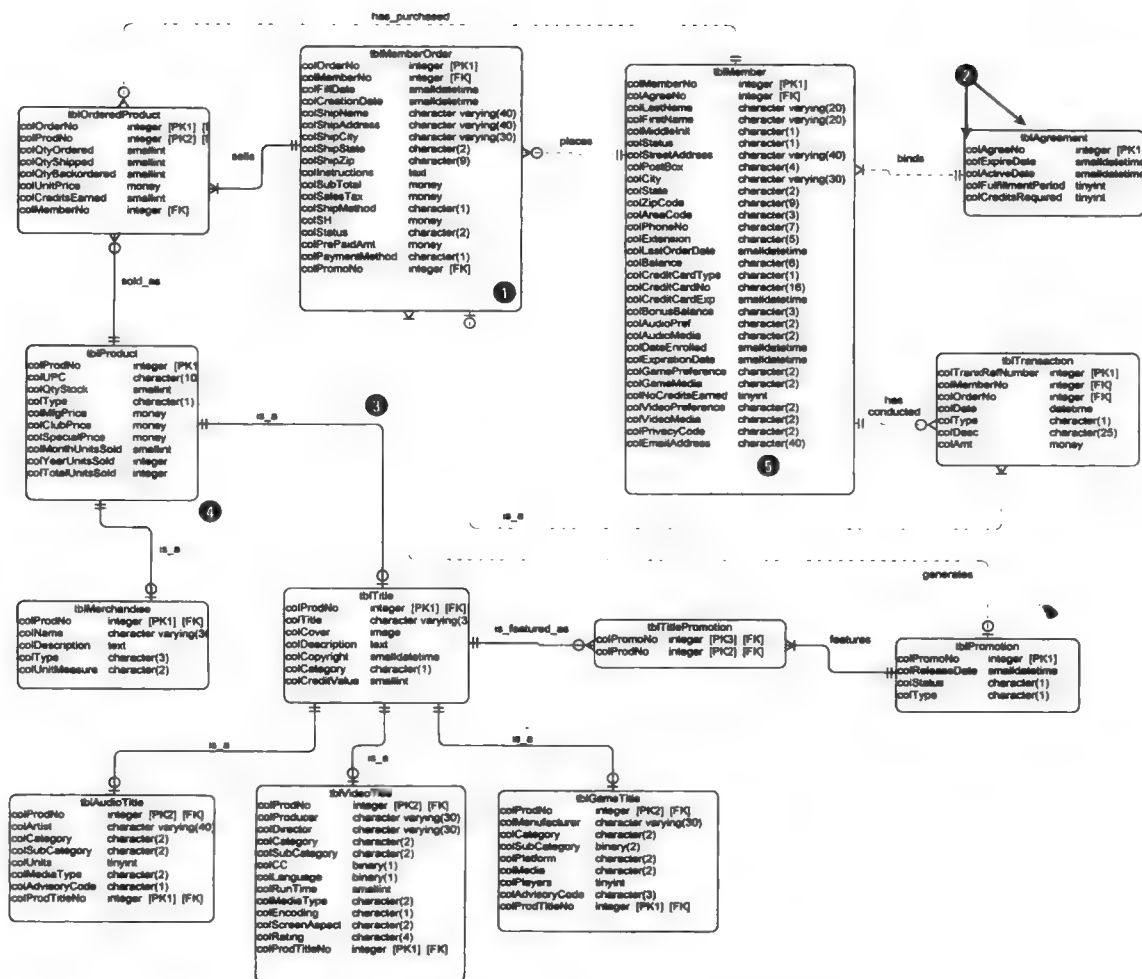


图 13-8 初始的音阶公司项目物理数据库模式

db 指示数据库自身。

idx 指示为表构建的一个索引。

dom 指示可以应用于一个或多个字段的字段范围。

③ 逻辑关系（既包括确定性关系也包括非确定性关系）被转换成使用外键实现的约束。

① 我们选择把逻辑泛化层次中的每个超类和子类实体都实现成物理表（这是 System Architect 的物理数据模型生成器默认的设置）。

⑤ 注意 System Architect 自动为每个字段推测了物理数据类型，这是根据：1）选择微软公司的 SQL Server 作为目标数据库管理系统；2）在系统分析期间为每个实体属性定义的逻辑数据类型。可以修改生成的物理数据类型，以减少所需的存储空间、提高数据完整性或者更好地表示包含在字段中的所有可能值。

尽管在数据库模式中没有描述，但模式生成器还为模式中指示的每个主键创建了一个索引。可以为唯一的次键（例如 tblProduct 中的统一产品编号或 UPC 字段）增加额外的索引，或者为任何可以用子集来划分表中所有记录的非键属性（例如 tblTransaction.colType）增加索引，这些索引可以提高最终数据库的性能。

有些 CASE 工具生成的数据库模式比我们的例子更详细。例如，有些数据库模式为每个字段指示了该字段是否必须值：

- NULL 意味着该字段不必有值。
- NOT NULL 意味着该字段必须有值。因为主键用于唯一地访问记录，所以 PK 字段不能取 NULL 值。

当设计数据库时，你曾经想过合并第三范式实体吗？例如，你曾经想过把两个第三范式实体合并成一个表（一般那将不再是一个第三范式实体）吗？通常不要这样做。尽管数据库管理员可以创建这样一个合并方案以提高数据库性能，但他必须仔细地权衡其优缺点。虽然这样的合并方案可能意味着极大的方便，由于拥有更少的表和更好的整体性能，这种组合也可能会造成数据独立性的损失，如果未来新的字段需要重新将这个表分割成为两个表，则必须重写程序。作为一条通用规则，不推荐将实体组合成一个表。

13.3.3 数据完整性和访问完整性

数据库完整性是有关信任的问题。企业及其用户可以信任存储在数据库中的数据吗？数据完整性为数据库提供了所需的内部控制，任何数据库中都必须至少设计三类数据完整性。

13.3.3.1 键完整性

每个表都应该有一个主键（可以是复合键）。必须控制主键以使表中没有两个记录具有相同的主键值（注意对于复合键，组合值必须唯一——构成组合的单个键值不必唯一）。

而且，绝不允许记录的主键具有 NULL 值，那将违背主键的初衷：主键是用来唯一地确定一个记录的。

如果数据库管理系统没有强制执行这些规则，那就必须采取其他措施确保这些规则，绝大多数 DBMS 确实强制执行了键完整性。

13.3.3.2 域完整性

必须设计合适的控制以确保每个字段都有合法值。例如，如果 GRADE POINT AVERAGE 字段定义为 0.00 ~ 4.00 之间的数，那么必须实现控制以防止出现负数和大于 4.00 的数。

不久以前，用户期望应用程序进行所有的数据编辑工作；但如今，大多数数据库管理系统能够强制执行域完整性规则。在可以预见的未来，数据编辑的任务将继续由应用程序和 DBMS 共同分担。

13.3.3.3 访问完整性

关系数据库架构通过外键在记录之间实现了关系。外键的使用增加了数据库的灵活性和可扩展性，但它也增加了访问完整性错误的风险。当一个表中的一个外键值在相关表中没有匹配的主键值时，就存在访问完整性错误。例如，INVOICES 表通常包括外键 CUSTOMER NUMBER 来“索引回”CUSTOMERS 表中相匹配的主键 CUSTOMER NUMBER。如果删除一个 CUSTOMER 记录会发生什么情况？有可能存在某个 INVOICE 记录，而在 CUSTOMERS 表中没有同这个记录的 CUSTOMER NUMBER 字段匹配的记录。从根本上说，在两个表中包含了访问完整性。

如何防止出现访问完整性错误？可以采取两种措施：当考虑删除 CUSTOMER 记录时，要么可以自动删除匹配这个记录的 CUSTOMER NUMBER 字段的所有 INVOICE 记录（它们不再有多少业务意义），要么不允许删除 CUSTOMER 记录，除非已经删除了所有相关的 INVOICE 记录。

访问完整性以删除规则的形式说明如下^①：

- 没有限制——可以删除表中的任何记录而不用考虑其他表中的任何记录。

在看了最后的音阶公司案例的数据模型后，我们不能对任何表应用这条规则。

- 删除：瀑布式——删除表中的一个记录必须自动地紧跟着删除相关表中的匹配记录。许多关系型 DBMS 可以使用触发器自动地强制执行“删除：瀑布式”规则。

在音阶公司案例的数据模型中，有效的“删除：瀑布式”规则的例子包括 MEMBER ORDER 表和 MEMBER ORDERED PRODUCT 表。换句话说，如果我们删除一个特定的会员订单（MEMBER ORDER）记录，应该自动地删除与那个订单匹配的所有会员订购的产品（MEMBER ORDERED PRODUCTS）记录。

① 数据库知识丰富的学生知道对于访问完整性还有插入规则和修改规则，对这些规则的全面讨论在数据库课程和课本中讲授。

- 删除：限制式——不允许删除表中记录，除非与之匹配的记录已经从相关的表中删除。许多关系型 DBMS 可以自动地强制执行“删除：限制式”规则。

例如，在音阶公司案例的数据模型中，我们可以说明，只要还存在那个产品的会员订购的产品（MEMBER ORDERED PRODUCTS）记录，就不应该允许删除任何产品（PRODUCT）记录。

- 删除：设置空——表中一个记录的删除必须自动地紧跟着将相关表中任何相匹配的键值设置为 NULL。许多关系型 DBMS 可以通过触发器自动地强制执行这条规则。

“删除：设置空”规则没有用在音阶公司案例的数据模型中。只有在愿意删除一个主表记录而又不想删除对应的事务表记录（为了保存历史数据）时，这条规则才有用。通过设置外键为 NULL，被告知那个记录没有指向一个对应的主记录，但至少不会把它指向一个不存在的主记录。

最终的数据库模式（完成了访问完整性规则）在图 13-9 中显示。这是用来编写创建表和数据结构的 SQL 代码（或其等价物）的蓝图。

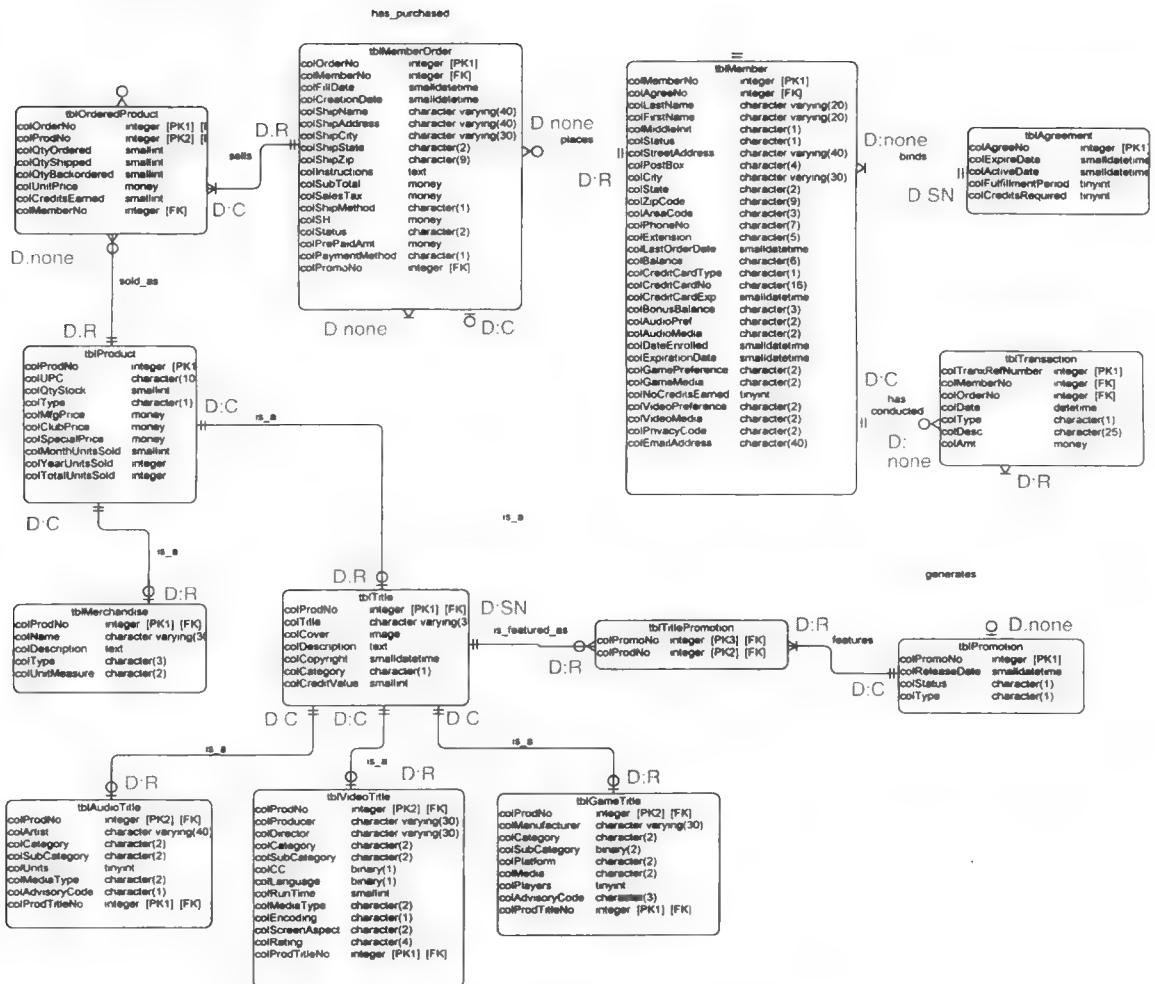


图 13-9 最终的音阶公司物理数据库模式

13.3.4 角色

有些数据库标准坚持两个字段的名称不能完全相同，这种约束简化了文档——有助于系统说明和元数据定义，但这与外键有一个明显的矛盾。按照定义，外键必须有一个对应的主键。在逻辑数据建模期间，使用相同名称与我们的目标是一致的，这有助于用户理解这些外键用来匹配在不同实体中的相关记录。但是在物理数据库中，并不总需要（或者甚至不期望）数据库中的这些冗余的字段名称。

为了解决这个问题，可以给外键设置角色名称。角色名称是外键的一个替代名字，它清楚地区分了外键在表中的用途。例如，在音阶公司实例的数据库模式中，PRODUCT_ NUMBER 字段是 PRODUCTS 表的一个主键，也是 MEMBER ORDERED PRODUCTS 表的一个外键。在 PRODUCTS 表中这个名称不应该改变，但是应该把 MEMBER ORDERED PRODUCTS 表的外键重命名为 ORDERED_ PRODUCT_ NUMBER，以更准确地反映它在 MEMBER ORDERED PRODUCTS 表中的角色。

是否使用角色名称的决策通常由数据管理员或数据库管理员做出。

13.3.5 数据库分布和复制

在第7章中，简要地介绍了逻辑数据分布分析的概念。数据分布分析确定哪些业务地点需要访问哪些逻辑数据实体和属性。

我们使用第7章介绍的简单矩阵将实体和属性映射到不同地点，许多 CASE 工具（包括 System Architect）都包含了构建这样一个矩阵的工具。现在应该考虑的是数据分布分析对数据库设计的影响。

在如今多层客户/服务器以网络为中心的世界中，信息系统和数据库很少是集中式的。相反，它们分布在一个可能跨越许多建筑物、城市、州或者国家的网络上。因此，可能需要将一个数据库设计的全部（或部分）分割、分布或复制到不同物理地点的不同物理数据库服务器上。从根本上说，需要进行物理数据库分布分析，这种分析考虑了我们在逻辑数据分布分析期间了解到的知识。

可以使用以下一些基本的分布方式：

- 数据库的集中。换句话说，不管需要访问数据库的物理地点有多少，都将在单个服务器上实现数据库。这个方案比较简单，也最容易维护；但是，对数据管理员和用户来说，它违反了一条非常重要的数据管理规则——数据应该放在尽可能同用户接近的地方。
- 数据的水平分布。在这种方式下，每个表（或表中的整个行）将被分配到不同的数据库服务器和地点。这种方式能够产生有效的访问和安全性，因为每个地点只拥有该地点需要的那些表和行。但是，对于需要综合多个站点数据的管理分析，将数据重新组合起来并不总是那么容易。
- 数据的垂直分布。在这种方式下，表中特定的列被分配到特定的数据库和服务器上，其优缺点与水平分布方式的优缺点十分相似。
- 数据的复制。复制是指在多个地点物理地重复整个表。大多数高端企业级数据库管理系统都支持复制技术，它们协调对复制的表和记录的修改，以维护数据完整性。这个方案具有性能高和可访问性好的优点，并减少了网络流量，但是它也增加了数据完整性的复杂度，而且需要更多的物理存储空间。这些分布方式之间不是互斥的，设计人员必须仔细地规划数据分布和复制的程度。

根据物理数据模式，我们就可以定义对应特定地理位置的视图（以及对应不同用户和应用的子视图）。数据库视图可以具有很大的选择性，它可以包含表的特定子集、表中列的特定子集甚至表中记录的特定子集。每个视图必须仔细地与主数据库模式保持同步，以使对主模式的修改可以（如果合适的话）传播到视图中。CASE 工具在定义视图和保持视图同步方面是很有用的。

对于音阶公司案例项目，我们计划在3个城市复制整个数据库，公共表的数据完整性将使用 SQL Server 的复制技术实现。系统分析员一般不编制复制规则，而由资深数据库分析员或管理员负责。既然我们将在每个城市的服务器上实现整个物理数据库模式，那就没有必要为项目定义视图了。

13.3.6 数据库原型

原型不是用来代替全面的数据库模式的。一旦完成了数据库模式设计，通常可以快速地生成一个原型数据库。大多数现代的 DBMS 都包含了功能强大的菜单驱动式数据库生成器，它可以自动地创建一个 DDL，并从该 DDL 生成一个原型数据库，然后这个数据库可以加载测试数据，这对于原型化和测试输出、输入、屏幕以及其他系统组件是很有用的。

13.3.7 规划数据库容量

数据库存储在磁盘上。最终，数据库管理员将要为新数据库估计磁盘容量，以确保有足够的磁盘空间可用。数据库容量规划可以使用如下简单的算法计算，这个简单的算法忽略了一些因素，例如封

装、编码和压缩，但通过忽略这些因素，也就留出了一定的空间余量。

1. 对于每个表，求和字段大小，得到的就是表的记录大小。不要考虑压缩、编码和封装的影响——换句话说，假设每个存储的字符和数字都将消耗一个存储字节。注意格式化字符（例如，逗号、分号、连字符等）几乎从来不存储在数据库中，这些格式化字符是由将访问数据库并向用户表现输出的应用程序加上的。
2. 对于每个表，记录大小乘上要被包含在表中的实体实例数量，建议考虑一段合理时间段的增长（例如3年），得到的就是表大小。
3. 求和表大小，得到的就是数据库大小。
4. 作为备选，加上一个空间余量缓冲（例如10%），以考虑未预期的因素或者上面的不正确估计，得到的就是预期的数据库容量。

13.3.8 数据库结构生成

CASE 工具经常能够直接从基于 CASE 的数据库模式为数据库生成 SQL 代码，这个代码可以输出到 DBMS 进行编译。甚至一个小型数据库（例如音阶公司案例的模型）就可能需要 50 页或者更多的 SQL 数据定义语言代码，用来创建表、索引、键、字段和触发器。显然，CASE 工具自动生成语法正确代码的能力是一个很大的优点。进一步说，修改数据库模式并重新生成代码几乎总是比直接维护代码要容易。

复习题

1. 缩写 CRUD 代表什么？
 2. 关于数据库常见的错误概念是什么？
 3. 为什么在数据库中存储数据比在文件中存储数据风险更高？
 4. 什么是次键？
 5. 什么是固定长度记录结构？
 6. 有哪些常见类型的常规文件和表？
 7. 在比较运行数据库和数据仓库时，哪个通常具有
- 较少的 CRUD 活动？为什么？
 8. 什么是数据库引擎？
 9. 什么是元数据？如果数据库管理员需要定义元数据，应该使用哪种语言（DDL 还是 DML）？为什么？
 10. 什么是关系数据库？
 11. 关系数据库模式和数据库模式之间有什么区别？
 12. 用于强制访问完整性的公共删除规则是什么？

问题和练习

1. 本书提到“数据是必须被控制和管理的资源”。解释这句话，指出你同意与否，为什么？
 2. 考虑一个地区小汽车代理商，他已经经营了 15 年或者更长时间了。除了销售和出租新的和用过的汽车，这个代理商还有一个零部件部门、服务部门和汽车团体部门。它是 70 年代中期首个汽车代理商之一，而且大多数信息系统已经被周期性地更新过，少数原先的系统仍在使用，包括一些独立应用系统。大约 5 年前，代理商开始逐步地将一些传统的基于文件的系统替换成关系数据库系统，但替换过程远没有结束。去年，代理商在一本航空杂志上读到关于数据仓库的信息，并雇用了一个本地集成商安装了一个数据仓库，该项目仍在进行中。代理商有一堆万事通的 IT 职员，但大多数开发工作是外包出去的。给定这个场景，绘制一张高级视图说明该代理商的数据架构可能的样子，以图 13-1 为例。
 3. 习题 2 中代理商的遗留的独立应用系统之一是销售员工作调度系统。这个系统是 80 年代在一台
- 微机上使用 dBASE III 开发的，用来跟踪每个销售人员每日和每周的工作进度。代理商计划让这个古董系统退休，并引入一个关系数据库系统。你可能期望在一个调度系统中找到哪些表？以图 13-4 为例，在一个物理数据库模式中展示这些表。每个对象包括主键、任何外键、一个或两个非键属性，以及一些值。

 4. 匹配下面第 1 列中的词汇和第 2 列中的定义或者例子。
- | | |
|------------|---------------------|
| 1. DBMS | a. 数据库的物理实现 |
| 2. 事务文件 | b. 国家代码的联邦公报 |
| 3. 数据仓库 | c. 5NF |
| 4. 主键 | d. Microsoft Access |
| 5. SQL | e. Delete: Set null |
| 6. CASE 工具 | f. Sybase IQ |
| 7. 匈牙利命名法 | g. SSN |
| 8. 规范化 | h. 每日医院记录文件 |

(续)

9. 表查询文件	i. Oracle 10g
10. 访问完整性	j. 表的标准命名规范
11. 模式	k. System Architect
12. 个人数据库	l. ALTER TABLE

5. 你是一名系统设计人员，你有一个朋友，他拥有一个书店，专门邮购稀有书籍和首版书。商店和邮购加在一起，每天平均销售大约 12 本书。你的朋友也想在 Web 网上开始售书，并且找你提供建议。他听说他应该使用一个关系数据库，Oracle 是最好的。你认同吗？如果不认同，你将推荐什么？
6. 在将逻辑数据模型转换成物理数据模式时，数据库管理员在超类/子类实体如何实现方面有哪些选择？
7. 请计算构造一个具有如下 4 张表的数据库的容量：

表 1	表 2	表 3	表 4
字段 1: 32 字符	字段 1: 5 字符	字段 1: 2 字符	字段 1: 16 字符
字段 2: 15	字段 2: 7	字段 2: 7	字段 2: 30
字段 3: 7	字段 3: 13	字段 3: 8	字段 3: 12
字段 4: 12	字段 4: 6	字段 4: 4	字段 4: 7
字段 5: 9	字段 5: 4		字段 5: 54
字段 6: 12			字段 6: 3
字段 7: 6			字段 7: 1
字段 8: 2			

项目和研究

1. 时常，一个企业的数据库环境反映了经过多年发展起来的数据结构，有时偶然地，经常反映了一种架构风格和结构。看看你的企业或学校或者一个本地公司的现有的数据库环境。
- a. 企业中使用的最老的信息系统有多长时间？
 - b. 最新的信息系统有多长时间？
 - c. 企业既有传统的基于文件的系统，也有现代的关系数据库系统吗？
 - d. 有哪些不同的用户系统、最终用户工具、最终用户应用？有数据仓库吗？有基于 Web 的应用系统吗？
 - e. 基于图 13-1 显示的图绘制一个数据架构图。
2. 现代数据库需要高级的技巧和知识以充分地支持它。你是否能同 3~4 个本地企业的数据库管理员交谈，了解他们使用现代的数据库系统：
- a. 描述同你交谈的数据库管理员管理的系统。
 - b. 一般地，在他们成为数据库管理员之前，他

表 1 初始加载 200 000 条记录，表 2 加载 100 000 条记录，另外两张表每个 40 000 条记录。在记录数上的预期增长率是 3 年每年 20%。

按照本章描述的数据库容量规划步骤，预期的数据库容量是多少？

8. 你正在研究你的企业使用的一个十分大的信息系统的设计文档。如预期的，逻辑数据模型中的所有实体都是三阶范式的。但在比较逻辑数据模型和物理数据库模式中，你注意到在逻辑数据模型中独立的两张表在物理数据库模式中组合成了一张表。没有注释解释这一点，你也无法询问签署了设计文档的数据库管理员，因为他已经不在公司了。折中三阶范式可能的原因是什么？潜在的后果是什么？
9. 解释访问完整性的概念，举一个例子。什么是访问完整性错误？举例说明，并解释访问完整性错误可能的后果。
10. 用于强制访问完整性的删除规则包括 Delete: Cascade 和 Delete: Restrict。通常，DBA 应该使用什么评价规则来决定是使用 Delete: Cascade 还是 Delete: Restrict 进行删除？
11. 数据库集中是分布方案之一，但它违反了数据应该在接近其用户的地方管理和存储的规则。讨论为什么许多数据库管理员和用户认为这条规则很重要。随着基于 Web 的应用和全球信息系统的增长，这条规则仍然重要并且/或者可行吗？

们需要拥有多少信息系统方面的经验和培训？

- c. 一般地，每年他们花多少时间用于培训（正式的或非正式的），以保持他们的技术时新？他们觉得他们得到了足够的培训吗？
 - d. 计算每个数据库管理员年度培训的平均费用。包括直接的培训费用和间接的“机会”成本。如果实际的费用不可知，使用直接培训费用每天 500 美元，DBA 平均工资每小时 75 美元进行计算。
 - e. 你认为 DBA 管理员的培训费用高于平面文件系统管理员的培训费用吗？为什么？
3. 当你同数据库管理员讨论关于培训的内容后，再询问他们以下问题：
- a. 规范化——他们通常会规范化到三阶范式吗？更高的范式呢？
 - b. 一般地，一个大型的或者企业级的数据库需要花费多长时间进行规范化？

- c. 每个数据库管理员面对三个最主要的问题是什么?
 - d. 他们要多频繁地修改和/或更新数据库模式? 他们以正式的过程进行更新吗, 还是以某种随意的方式进行?
 - e. 如果他们是企业的 CIO, 他们会改变什么?
4. CASE 工具, 例如 System Architect, 用于进行数据库开发和维护。在网上和商业期刊上查找一些当前使用的流行的 CASE 工具的信息。
- a. 你找到了哪些 CASE 工具, 它们是谁生产的?
 - b. 使用这些 CASE 工具的数据库或 IT 商店的数量?
 - c. 这些 CASE 工具的费用在什么范围? 你认为它们值这个价吗?
 - d. 按照它们的特点和功能比较这些 CASE 工具。
 - e. 如果你是一个 DBA 而且不用考虑费用问题,

你会使用哪个? 为什么?

5. 目前, 关系数据库技术可能是现代信息技术中使用最流行的数据库技术。但数据库技术是一个不断发展进化的领域, 新的技术被不断地开发。在网上或者你学校的图书馆查找关于正在出现的数据库技术的文章, 例如对象数据库管理系统。注意要包括来自公司的白皮书, 例如 Oracle、Sybase、IBM 和 Microsoft。
 - a. 你找到了什么文章?
 - b. 有什么新的数据库技术正在进入市场, 或者正在开发之中?
 - c. 比较这些技术和现代的关系数据库技术。
 - d. 因特网技术的增长对这些技术有什么影响?
 - e. 如果存在的话, 你认为哪一个是能够代替关系数据库技术的最重要的竞争者? 为什么?

小型案例

在老师的帮助下, 你同来自另一个学校的系统分析班级或者计算机编程班级的团队联络。你的任务(要一起完成)是为你选择的一个小型企业或非营利的组织构建一个合适的 Web 页面。对你们的完成情况、功能、职业性和团队工作方面进行打分。通信方面的建议是尽可能地使用电子邮件。

1. 通过电话、虚拟会议环境、讨论板或者电子邮件同另一个学校的团队会面。如果你使用虚拟会议环境, 你可能需要安装和学习如何使用合适的软件。确定和建立团队指南规则, 为了:
 - 最终期限——团队将如何处理由于某个团队成员造成的进度延误? 谁将负责设置时限?
 - 通信——你们将如何交流? 多频繁? 你的一些成员使用某种方法交流得比其他方法更好吗(例如, 更偏好某种方法)?
 - 个人差异导致的错误交流——你将如何涉及会员之间产生的错误交流或者导致的个体差异?
 - 预期——团队对质量的预期是什么? 行为呢?

团队和个人练习

1. 使用到目前为止你学到的词汇和概念创建一个填字游戏。然后, 与同学们交换填字游戏。你们中的每个人要完成别人的填字游戏。
2. 个人练习: 在本章的小型案例中, 你与来自另一个学校的团队一起完成了一个网站项目。用一篇短文总结你的经验, 并提交给你的老师。

如果某人没有做到预期的效果你将如何处理? 给你的老师提交一份备忘录, 由每个团队成员签名, 内容涉及这些情况将如何解决。

2. 同本地的小型企业所有者或者非营利组织的代表会面。寻找一个愿意接纳你的团队为他们制作一个网站(免费的)的企业或组织。从你的学校的风险管理或法律部门找出你和你的“客户”需要完成哪些手续(为什么这些手续是必需的)。
3. 通过交谈、调查表、调研、JAD 之类的工具确定企业或组织的需求, 为网站创建合适的模型。在你的模型和论文中不要忘了考虑费用、法律问题和公司的特殊需求。将按照完整性、正确性、清晰度和专业性方面对你进行评分。
4. 使用恰当的技术创建网站, 获得一个域名等。使用域名建立电子邮件。如果公司需要购物车和联机支付功能, 确保这些功能是完整的。在你将它提交给你的老师或客户之前, 通过同其他的团队交换 URL 进行站点强度测试。

3. 阅读你的本地黄页电话本。在你的地区谁是“系统分析和设计”的主要推广者? 至少同其中的三个联系, 找出它们提供什么服务, 它们的专长/经验, 以及它们的收费标准? 这些信息对你和你的职业来说意味着什么? 把你的结论拿到课堂上讨论。

输出设计和原型化

本章概述和学习目标

在本章中，你将学到如何设计计算机输出，具体包括以下内容：

- 区分内部输出、外部输出和回转输出。
- 区分详细报告、总结报告和例外报告。
- 确定几种输出实现方法。
- 区分用于表现信息的表格格式、分区格式和图形格式。
- 区分面积图、条形图、直方图、饼图、线条图、雷达图、环形图和散点图及其应用。
- 描述几个重要的输出设计通用原理。
- 设计并原型化计算机输出。

输出和输入设计代表了某种类似“鸡和蛋”的问题：你先做哪一个？本书首先介绍输出设计。经典的系统设计称这种方式为系统验证测试——设计输出，然后确定足够的输入以产生输出。在实践中，设计输出和输入的顺序不那么重要，因为现代系统分析技术已经充分地预先定义了逻辑输入和逻辑输出需求。如果愿意的话，可以随意地改变第 14 章和第 15 章的学习顺序。

本章关键术语

内部输出（internal output）是提供给系统所有者和组织内的系统用户的输出。

详细报告（detailed report）是显示很少（或者没有）经过过滤的信息的内部输出。

总结报告（summary report）是为管理人员分类信息的内部输出。

例外报告（exception report）是过滤数据以报告对某些情况或标准的例外信息的内部输出。

外部输出（external output）是离开组织的输出。

回转输出（turnaround output）是指可能重新进入系统作为输入的外部输出。

表输出（tabular output）是以文本和数字列的形式表现输出。

分区输出（zoned output）是把文本和数字放置在一个表格或屏幕的指定区域或方框中的输出。

图形输出（graphic output）是使用图片化图表来揭示信息的输出。

14.1 输出设计概念和指南

输出向系统用户表现信息。输出是一个运行中的信息系统最可见的部分。因此，它们通常是用户和管理层最终评估系统价值的基础。在需求分析过程中，定义了逻辑输出需求；在决策分析过程中，可以考虑不同的物理实现方案。在本章中，将学到如何实际地设计输出。

如今，大多数输出都通过快速构造的原型来设计。这些原型可以由计算机产生的简单且具有虚拟数据的实体模型，或者可以从原型数据库（例如 Access，它可以快速地用测试数据构造原型）中产生的程序。这些原型一般功能不完整，它们没有包含最终版本的系统中所需的安全特征和数据访问优化。此外，由于对设计效率的考虑，可能没有包括那些应该包含在实际系统中的每个按钮或控制特征。

在需求分析过程中，输出被建模成包含数据属性的数据流。即使在最全面的需求分析中，也可能会遗漏某些需求，所以输出设计也可以将新的属性或域引入到系统中。

我们从输出类型的讨论开始。输出可以按照两个特性进行分类：1）它们的分布和观众；2）它们的实现方法。图 14-1 说明了这个分类法，这些特征将在下面的小节中简要介绍。


分布 实现方法	内部输出 (报告)	回转输出 (先外部后内部)	外部输出 (事务)
打印机	打印在硬拷贝上供内部使用的 详细信息、总结信息或例外信息 常见范例：管理报告	打印在业务表格上的业务事 务，这些业务表格最终会被返回 作为输入业务事务 常见范例：电话账单和信用卡 账单	打印在业务表格上的业务事务， 这些业务表格总结了业务事务 常见范例：工资单和银行结算表 账单
屏幕	显示在显示器上供内部使用的 详细信息、总结信息或例外信息 报告可以是表格形式，也可以是 图形形式 范例：联机管理报告和对查询 的响应	显示在显示器上的表单和窗口 中的业务事务，表单和窗口还将 用来输入数据，以激发另一个相 关事务 范例：可以单击“购买”选项 的股票价格的 Web 显示	显示在业务表格上的业务事务， 这些业务表格总结了业务事务 范例：基于 Web 的详细银行事务 报告
零售终端	打印或显示在具体内部业务功 能的专用终端上的信息。包括无 线通信信息传输 范例：换班时的现金出纳机结 算报告	打印或显示在专用终端上的信 息，用于激发一个后续的事务 范例：杂货店监视器允许用户 浏览通过借贷卡进行自动付款的 产品价格	打印或显示在面向客户的专用终 端上的信息 范例：账号余额显示在 ATM 机上 或者打印输出；另外，账号信息通 过有线电视或者卫星电视显示
多媒体（音频 或视频）	被转换成语音供内部用户使用的 信息 对于内部用户来说并不常用	被转换成语音供外部用户使用的 信息，外部用户用语音或者按 键输入数据 范例：作为课程注册系统一部 分的电话按键音课程调度	被转换成语音供外部用户使用的 信息 范例：提供给潜在的 DVD 联机购 买者的电影片断，或者抵押支付查 询的电话响应
电子邮件	关于内部业务信息的显示消息 范例：宣布新的联机业务报告 可用的电子邮件消息	用来启动业务事务的显示消息 范例：要求回复以继续处理业 务事务的电子邮件消息	关于业务事务的消息 范例：通过 Web 上的电子商务进 行业务交易的电子邮件消息证实
超链接	到内部信息的 Web 链接，采 用 HTML 或 XML 格式 范例：所有信息系统报告集成 为一个基于 Web 的档案系统， 供联机访问	包含在 Web 输入页面中的 Web 链接，给用户提供了到其他信息 的访问 范例：在一个 Web 拍卖页面 上，到销售商销售历史的链接， 并邀请增加新评价	包含在 Web 事务中的 Web 链接 范例：到保密策略的超链接，或 者到解释或响应一个报告或事务 的信息链接
微缩胶片	内部管理报告归档到只要求很 小物理存储空间的微缩胶片上 范例：计算机微缩胶片输出 (COM)	不可应用，除非内部需要归档 回转文档 范例：计算机微缩胶片输出 (COM)	不可应用，除非内部需要拷贝外 部报告 范例：计算机微缩胶片输出 (COM)

图 14-1 计算机生成的输出的分类法

14.1.1 输出的分布和观众

分类输出的一种方法是按照其在组织内和组织外的分布以及读取和使用它们的使用者进行分类。内部输出面向内部系统所有者和组织内的系统用户，它们很少用于组织外部。内部输出要么支持日常的业务操作，要么支持管理监视和决策制定。图 14-2 说明了内部输出的 3 个基本子类，分别如下。

- 详细报告显示很少（或者没有）经过过滤和限制的信息。图 14-2a 中的例子列出了在某个特定日期生成的所有购买订单，其他详细报告的例子是库存中所有客户账号、订单或产品的详细清单。有些详细报告是历史性的，有一些详细报告则是规定的，即是政府要求的。
- 总结报告为不想费力阅读细节的管理人员分类信息。图 14-2b 的例子报告按照产品类型和类别总结了月度和年度总销售情况。总结报告的数据一般被分类和总结，以支持分析趋势和潜在问题。总结报告中图表和图形的使用也越来越普及，因为它更直观清楚地总结了趋势。
- 例外报告在作为信息表现给管理人员之前过滤数据，它仅包含对某些情况或标准的例外信息。图 14-2c 的例子描述了拖欠债务的会员账号身份，例外报告的另一个典型例子是确定低库存产品的报告。



SoundStage Entertainment Club
Fax 317-494-5222

PURCHASE ORDER

The following number must appear on all related correspondence, shipping papers, and invoices:
P.O. NUMBER: 712612

To:
CBS Fox Video Distribution
26253 Rodeo DR
Hollywood, CA

Ship To:
SoundStage Entertainment Club
Shipping/Receiving Station
Building A
2630 Darwin Drive
Indianapolis, IN 45213

P.O. DATE	FROM/TO/ORDER	SHIP VIA	A.O.B. DATE	TERMS
5-3-06	LDB	UPS		Net 30

QTY	DESCRIPTION	UNIT PRICE	TOTAL
20000	Star Wars: The Phantom Menace (VHS)	15.99	319,800.00
3000	Star Wars: The Phantom Menace (DVD Dolby Digital)	19.99	59,970.00
500	Star Wars: The Phantom Menace (DVD DTS)	24.99	12,495.00
8000	Star Wars: The Phantom Menace (PlayStation II)	16.99	135,920.00
400	Star Wars: The Phantom Menace Soundtrack (CD)	16.99	6,796.00
600	Star Wars: The Phantom Menace Theater Poster	4.99	2,994.00

Subtotal

537,975.00

Tax

37,658.25

Total

575,633.25

1 Please send two copies of your invoice

2 Enter this order in accordance with the prices, terms, delivery method, and specifications listed above

3 Please notify us immediately if you are unable to ship as specified

Madge Worthy

5-4-06

Authorized by

Date

图 14-3 典型的外部文档

虽然可以在针式打印机上打印输出，但是越来越多地是在激光打印机上打印输出，激光打印已经变得越来越划算了。内部输出一般打印在空白纸张上（称为普通纸张），外部输出和回转文档则打印在预打印的表格上。预打印表格的布局（例如空白支票和 W-2 税收表格）是预先确定的，而且空文档可以大批量生产，打印机最终在预打印表格上打印各种业务数据（例如工资单和 W-2 税收表格）。

打印输出最常用的格式也许是表。表输出以文本和数字列的形式表现的输出。大多数你写过的计算机程序都可能要产生表报告，本章前面演示的详细报告、总结报告和例外报告（见图 14-2）也都是表。


一种特殊的表输出是分区输出。分区输出把文本和数字放置在一个表格或屏幕的指定区域或方框中。分区输出经常与表输出一起使用，例如：订单输出除了包含表示订单产品的表（或者列和行）以外，还包含表示客户和订单数据的区域。

14.1.2.2 屏幕输出

增长最快的计算机输出介质是在可视显示设备上联机显示信息，例如 CRT 终端或 PC 监视器。如今的经济发展的需要及时的信息，屏幕输出最适合这个需求。

虽然屏幕输出为系统用户提供了对信息的便利访问，但是信息只是暂时的。当信息离开屏幕时，信息就消失了，除非重新显示。因此，打印输出通常会被添加到屏幕输出设计中。

借助屏幕输出技术，表报告（特别是总结报告）可以用图形格式表现。图形输出是使用图片化图表来揭示信息，它演示了表输出中不容易看出的趋势和关系。



SoundStage Entertainment Club
 2630 Darwin Drive - Bldg B
 Indianapolis, IN 45213
 317 496 0998 fax 317 494 0999

Invoice No. 301231

Customer
 Name KATRINA SMITH
 Address 3019 DURAC DR
 City LITTLE ROCK State AR ZIP 42653
 Phone 502-430-4545

INVOICE
 Due Date 2/24/06
 Order No. 346910
 Payment Amt

Detach and return top portion with payment

Qty	Description	Unit Price	TOTAL
1	EAGLES HELL FREEZES OVER (DVD DD)	\$19.99	\$19.99
1	THE GRAMMY BOX (CD) ***COUNTS AS 3 CREDITS	\$21.99	\$21.99
1	GONE WITH THE WIND DIRECTORS CUT (DVD DS)	\$17.99	\$17.99
1	SIXTH SENSE (VHS)	FREE SS CR	\$0.00
1	A BUG'S LIFE (VHS)	FREE SS CR	\$0.00
1	NASCAR 2000 (VHS) *** CLOSEOUT (NO SS CR)	\$9.99	\$9.99

10 SOUNDSTAGE CREDITS WERE USED TO PAY FOR PART OF THIS PURCHASE

WE APPRECIATE THE FINE MANNER IN WHICH YOU HAVE PAID ON YOUR ACCOUNT. IN APPRECIATION WE HAVE ADDED 7 SOUNDSTAGE CREDITS TO YOUR ACCOUNT

YOU CAN EARN 7 CREDITS BY PAYING THIS INVOICE BY THE DUE DATE

Payment Details
☐ Cash
☐ Check
☐ Credit Card
 Name _____
 CC # _____
 Expires _____

SubTotal \$69.96
 Shipping & Handling \$7.00
 Taxes \$2.95
TOTAL \$79.91

Office Use Only

*Please return top portion invoice with payment Make checks payable to:
SoundStage Entertainment Club.*

RETURN TOP PORTION WITH PAYMENT

图 14-4 典型的回转文档

对于系统用户而言，一幅图比言语更更有价值。很多不同类型和风格的图表都可以用来表现信息，图 14-5 总结了现今的技术可以输出的各种类型图表。报告编写技术和电子表格软件可以快速地将表数据转变成图表，这些图表有助于阅读者更快速地得出结论。

物美价廉的图形打印机和软件也促进了图形输出的普遍应用，特别是在 PC 行业中。本章后面将显示音阶公司案例输出的一个图形设计。

14.1.2.3 销售点终端

许多如今的零售和客户交易由销售点 (point-of-sale, POS) 终端支持 (或增强)，典型的例子是自动取款机 (ATM)。POS 终端既包含输入设备，也包含输出设备。本章只介绍输出。ATM 显示账号余额，并打印事务收据；当条形码被扫描时，POS 收银机显示价格和总数，并生成收据；彩票 POS 终端生成随机数并打印彩票。所有这些例子都是必须被设计的输出。

14.1.2.4 多媒体

多媒体描述了传统的数字、编码和文字以外的信息表现，它包括图像、声音、图片和动画，通常是为对屏幕输出的临时扩充。信息系统应用向因特网和内联网的转变促进了多媒体输出的普遍应用。









类 型	例 子	选 用 准 则
线条图		线条图显示一段时间内一个或多个数据序列。它们用于总结和显示固定间隔数据，每条线代表了一个数据序列或者一类数据
面积图		面积图类似于线条图，但关心的焦点是线条下面的面积。图中的面积用来总结和显示数据随着时间的变化，每条线代表了一个数据序列或者一类数据
条形图		条形图用于比较数据序列或数据类，每个条形表示一个数据序列或一类数据
直方图		直方图类似于条形图，但条形是垂直的，而且一系列的直方图可以用来比较不同时间或不同时间间隔的同类数据，而每个条形代表了一个数据序列或者一类数据
饼图		饼图显示部分与整体之间的关系。用来总结在一个数据序列中数据占总体的百分比，图中的每一块代表了数据序列的一个数据项
环形图		环形图类似于饼图，但它们可以显示多个数据序列或数据类，每个数据序列或数据类都是一个同心环。在每个环中，环中的每一块代表了那个数据序列的一个数据项
雷达图		雷达图用来比较多个数据序列或数据类的不同方面，每个数据序列都被表示成围绕中心点的一个几何图形，多个序列互相重叠以便可以进行比较
散点图		散点图用来显示在不均衡的时间间隔内测量的两个或多个数据序列或数据类之间的关系，每个序列用不同颜色或记号的数据点表示

图 14-5 图表类型和选用准则

前面已经讨论了图形输出，但是其他多媒体格式也可以集成到传统的屏幕设计中。例如：许多信息系统提供电影和动画作为输出内容的一部分；产品描述以及安装和维护指南可以使用多媒体工具集成到联机产品目录中；声音也可以被集成。

但是多媒体输出并不局限于屏幕设计技术。声音（以基于按键电话系统的形式）可以用来实现一种有趣的输出方式：许多银行允许其客户使用按键电话访问各种账号、贷款和交易数据。

14.1.2.5 电子邮件

如果没有改变整个社会的话，电子邮件至少已经转变了现代企业的通信方式。可以预期，新的信息系统将是支持消息功能的。这对输出设计有什么影响呢？事务性系统越来越多地支持 Web 功能。当通过 Web 购买产品时，你几乎总是会收到自动的电子邮件输出，以确认你的订单，后续的电子邮件可能会通知你订单履行的进展，并要求客户反馈信息（回转输出的一种形式）。

内部输出也可以通过电子邮件得到增强，例如：系统可以把新报告可用的通知传送给感兴趣的用戶，于是只有那些真正需要报告的人将访问并打印报告。当需要大量分发信息时，这种方式可以明显地节省费用。

14.1.2.6 超链接

现在许多输出是通过 Web 实现的，许多数据库和客户订单系统也是支持 Web 的。Web 超链接使得用户可以按需浏览记录清单，或者查询特定记录并访问各种详细程度的信息。显然，这个介质也可以被扩展到计算机输入中。

有些技术可以将内部报告方便地转换成 HTML 或 XML 格式，并通过内联网分发，这减少了打印报告和屏幕报告对特定操作系统或操作系统版本（例如 Windows）的依赖性。从根本上说，接收者所需要的只是一个可以在任何计算机平台（Windows、Mac、Linux 或者 UNIX）上运行的浏览器。

但是 Web 输出不只是通过因特网和内联网表现传统输出。许多企业对基于 Web 的内部报告系统感兴趣，这种系统将传统的内部报告中的周、月和年度报告合并到一个有组织的数据库中，报告可以从中调出、显示或者打印。这些系统并没有创造新的输出，它们只是将以前的报告重新格式化供浏览器访问，所以可以把它看成是一个按需的 Web 报告归档系统。这种报告系统的例子有 DataWatch 公司的 Monarch/ES 和 NSA 公司的 Report Web。

14.1.2.7 微缩胶片

纸张体积大，故而需要相当大的存放空间。为了克服纸张的存放问题，许多企业使用微缩胶片作为输出介质。首选的胶片介质是微缩胶片，它们常被称为缩影胶片。一小块微缩胶片能够存储几十甚至上百页计算机输出。使用胶片也存在问题：微缩胶片和缩影胶片只能通过特殊的设备生产和阅读。

以上内容就是我们对输出概念介绍的全部内容。如果仔细地研究图 14-1，就会发现通过组合实现方法和分发方法可以开发出具有创造性的、用户友好的和令人激动的输出。

14.2 如何设计和原型化输出

在本节中，将讨论并演示输出设计过程；将介绍一些用于记录和原型化输出设计的工具，也将应用上一节中学到的概念；将演示系统用户和程序员如何使用自动化工具设计并原型化输出内容和输出布局。同往常一样，输出设计技术的每一步都将使用音阶娱乐俱乐部案例研究中绘制的例子来演示说明。

14.2.1 用于输出设计和原型化的自动化工具

在自动化工具出现以前，为了感觉系统用户期望输出是什么样子，分析员只能绘制输出草图。使用自动化工具可以开发出这些输出的更现实的原型。也许最便宜同时又最容易被忽视的原型化工具就是常用的电子表格软件，例如 Lotus 公司的 Lotus 1-2-3 和微软公司的 Excel。电子表格软件的表格格式特别适用于创建表格输出原型，而且大多数电子表格软件都具有快速地将表格化数据转换成各种流行图表格式的工具。因此，电子表格软件提供了一种经常被忽视的方式——能够快速地为系统用户设计图形化输出的原型。

最常用的输出设计自动化工具是 PC 数据库应用开发环境。许多读者无疑都在一门 PC 应用课程或者数据库开发课程中学过了 Access。因为在开发大多数企业级应用时 Access 还不够强大，所以你可能对如此多的设计人员使用 Access 进行原型化感到吃惊。首先，Access 提供了快速构造一个单用户（或者少量用户）数据库和测试数据的快速开发工具，那些数据以后可以送到输出设计原型中以增加现实性。设计人员可以使用 Access 的报告工具来布局建议的输出设计，并同用户一起测试它们。

许多 CASE 工具都包含了用于报告和屏幕布局以及原型化的工具，这些工具都使用需求分析期间创建的项目资料库。图 14-6 演示了 System Architect 的屏幕设计工具。

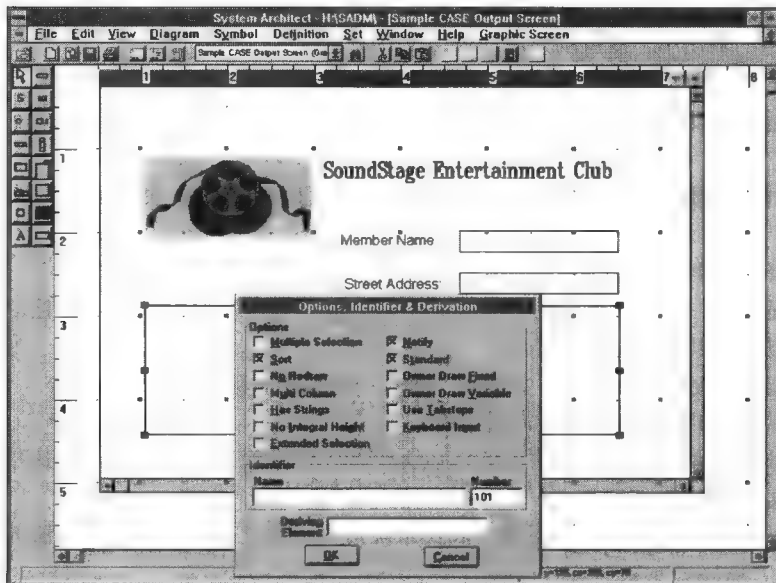


图 14-6 用于输出设计的 CASE 工具

以上自动化工具已经明显地加速并增强了输出设计过程。但是最终的输出设计过程将不只是设计输出的原型，而且还要设计输出的最后实现。这种较复杂的技术可以在报告编写工具中找到，例如 Seagate 公司的 Crystal Reports 和 Actuate 公司的 e. Reporting Suite，这些产品生成被集成到实际信息系统中的实际“代码”。图 14-7 演示了来自 Crystal Reports 工具的两个屏幕，我们用它从一个原型数据库中创建音阶公司案例的报告。

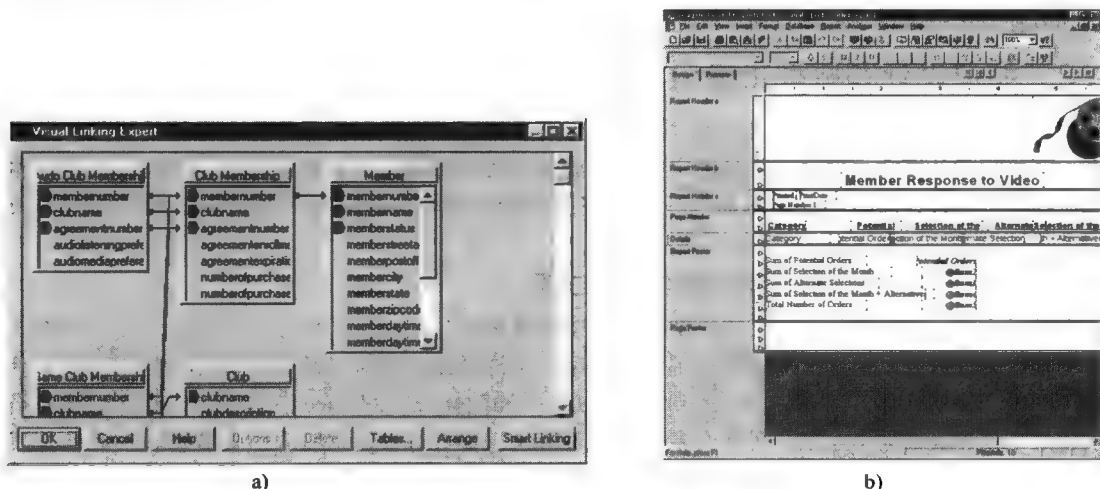


图 14-7 用于报告设计的报告编写工具

14.2.2 输出设计指南

输出设计中会遇到许多问题，大多数问题都是人类工程学方面的——希望设计出能支持系统用户工作方式的输出。以下是几个重要的输出设计原理：

1. 计算机输出应该易于阅读和理解。下面这些指南可以提高输出的可读性：
 - a. 每个输出应该有一个标题。
 - b. 每个输出应该标上日期和时间戳，这有助于读者掌握信息的时效性。
 - c. 报告和屏幕应该包括分段信息的节和标题。
 - d. 在基于表格的输出中，所有的字段应该清晰地标上标签。
 - e. 在基于表的输出中，列应该清晰地标上标签。
 - f. 因为节标题、字段名称和列标题有时被缩写以节省空间，报告应该包括或者提供这些标题的图例。
 - g. 只打印或显示需要的信息。在联机输出中，使用信息隐藏技术，并提供方法来扩展和减小信息的详细程度。
 - h. 信息从来不应该经过手工编辑才能使用。
 - i. 在报告或显示中，信息应该均匀分布——不要太拥挤，也不要太分散。而且，整个输出应该提供充分的边缘和空格，以提高可读性。
 - j. 用户必须能够方便地找到输出，方便地在报告中前移和后移，以及退出报告。
 - k. 输出中不应该出现计算机行话和错误消息。
2. 计算机输出的时效性很重要。输出信息必须在事务或决策需要信息之前到达接收者，这可能影响到如何设计和实现输出。
3. 计算机输出的分布（或访问）对所有相关系统用户必须是足够的。实现方法的选择影响到输出的分布。
4. 计算机输出对接收它们的系统用户来说必须是可接受的。一个输出设计可能包含了需要的信息，但仍不能被系统用户接受。为了避免这种情况发生，系统分析员必须理解接收者计划如何使用输出。

14.2.3 输出设计过程

输出设计过程并不复杂，有些设计步骤是基本的，而另一些则根据情况而定。在下面的小节中，将介绍这些步骤并演示音阶公司案例项目中的一些例子。

14.2.3.1 第 1 步：确定系统输出并检查逻辑需求

输出需求应该在需求分析过程中定义。物理数据流图（或者设计单元，二者在第 12 章都有描述）是输出设计的良好起点，DFD 既确定了系统的净输出（过程到外部代理的输出），也确定了输出的实现方法。

根据系统开发方法学和标准，每个净输出数据流也可以用数据字典或资料库（见第 9 章的数据结构）中的一个逻辑数据流描述，这个数据流的数据结构说明了包括在输出中的属性或域。如果那些需求以关系代数记号进行定义，你就可以快速地确定哪些域重复出现，哪些域具有可选的值，等等。考虑下面的数据结构：

数据结构定义逻辑需求	注 解
INVOICE = INVOICE NUMBER	←输出的唯一标识符
+ INVOICE DATE	←必须取一个值的多个域之一，没有括号说明需要有值
+ CUSTOMER NUMBER	
+ CUSTOMER NAME	←到相关定义的指针
+ CUSTOMER BILLING ADDRESS = ADDRESS >	
+ 1 { SERVICE DATE +	←开始重复调整 1 ~ n 次的域组合
SERVICE PROVIDED +	
SERVICE CHARGE { N	
+ PREVIOUS BALANCE DUE	←具有单个值的多个域
+ PAYMENTS RECEIVED	
+ TOTAL NEW SERVICE CHARGES	
+ INTEREST CHARGES	
+ NEW BALANCE DUE	
+ MINIMUM PAYMENT DUE	
+ PAYMENT DUE DATE	
+ (DEFAULT CREDIT CARD NUMBER)	
+ ([CREDIT MESSAGE PAYMENT MESSAGE])	←域不必有值
	←域不必有值，但如果有的话，它只能取两个可能值之一

如果没有这种精确的需求可用，则可以使用需求分析阶段创建的需求原型。无论如何，都应该以某种格式提供一个良好的需求陈述。

14.2.3.2 第 2 步：说明物理输出需求

决策分析阶段应该已经建立了对大部分输出数据流最终如何实现的一些预期。对于输出来说，要做出的决策是确定用于设计和实现的最佳介质和格式，这个决策基于以下内容做出：

- 输出的类型和目的。
- 运行可行性、技术可行性和经济可行性。

因为可行性不只对输出重要，所以评价可行性的技术在本书中单独讲授（见第 10 章）。但是，首要的评价准则描述如下：

- 输出供内部使用还是供外部使用？
- 如果是内部输出，它是详细输出、总结输出还是例外输出？
- 如果是外部输出，它是回转文档吗？

在理解了输出的类型及如何实现之后，需要考虑以下几个设计问题：

1. 什么实现方法最适用于输出？本章前面讨论了各种方法。你需要理解输出的目的或使用方式，才能确定正确的实现方法。可以为单个输出选择多种方法，例如：具有可选打印输出的屏幕输出。显然，这些决策最好由系统用户做出。

- a. 报告的最佳格式是什么？表格？分区？图形？还是这些格式的某种组合？
 - b. 如果想采用打印输出，就必须确定使用哪类表格或纸张。普通纸张有 3 种标准尺寸： $8\frac{1}{2}$ 英寸 \times 11 英寸、11 英寸 \times 14 英寸和 $8\frac{1}{2}$ 英寸 \times 14 英寸。你需要确定要使用的打印机的性能和限制。
 - c. 对于屏幕输出，你需要理解用户显示设备的限制。尽管更大的 19 英寸和 21 英寸的高分辨率的监视器越来越多，但大多数用户仍然使用 15 英寸和 17 英寸显示器，并把屏幕设置成 640×480 像素（特别是当你直接接触电子商务应用的客户时），所以我们仍然建议屏幕输出（包括应用中的表格和页）根据最低平均水平设计。
 - d. 表格图像可以存储下来，然后用激光打印机打印，从而消除了企业同表格生产厂家打交道的必要。
 2. 输出产生的频率如何？按需产生？每小时一次？每天一次？还是每月一次？对于计划的输出，系统用户什么时候需要报告？
 - a. 如果用户按需生成多份报告，那么一种有用的方法是使用自动化的电子邮件通知用户新的版本可用。
 - b. 如果报告由信息服务部门打印，它们就必须被安排到信息系统运行计划中。例如，一份系统用户在星期四上午 9:00 需要的报告可以被安排在星期四早晨 5:30 打印，其他时间都不太合适。
 3. 一个打印输出的单份拷贝会产生多少页输出？为了准确地计划纸张和表格消耗量，你可能需要这项信息。
 4. 输出需要多份拷贝吗？如果需要，多少？
 - a. 针式打印机通常用来同时打印一个多拷贝表格的所有拷贝。
 - b. 为了获得多份拷贝，激光打印机只能逐份地打印表格。这意味着如果拷贝在颜色和内容域上不同，预打印的表格必须在最终打印前校对好次序。
 5. 对于打印输出，分发控制定稿了吗？对于联机输出，需要确定访问控制。
- 这些设计决策应该在数据字典或项目资料库中记录。下面我们考虑音阶娱乐俱乐部案例的一个例子。

音阶公司的一个报告是：MEMBER RESPONSE SUMMARY REPORT，系统要求这份报告为内部管理人员提供关于客户响应月度促销商品的信息。设计人员确定了以下设计需求：

1. 经理将从他（或她）自己的工作站请求报告。已经决定了信息应该被表现为一个屏幕输出，以表格形式和图形形式表示（格式是通过原型决定的）。
 - a. 所有的经理都有 17 英寸或者更大的显示器。
 - b. 经理应该可以通过局域网使用激光打印机进行输出。打印输出应该使用 $8\frac{1}{2}$ 英寸 \times 11 英寸的普通纸张。
2. 经理必须能够按需显示报告。经理已经要求使用自动电子邮件通知任何新近生成的报告版本的可用性。到最新版本报告的超链接也应该在每个会员服务部经理（级别 3 和以上）的标准主页上显示。
3. 图形输出应该可以在单个屏幕上显示，并且可以在单页纸上打印；表数据可以在一页或两页纸上打印，通常认为页的数量对这份报告来说不重要。
4. 限制只有网络账号具有第 3 级或者更高特权的经理可以访问报告。报告应该包括一个“密级”水印以及一条提示消息，提示消息指出没有内部审计的书面许可禁止外部分发或者共享报告。

14.2.3.3 第 3 步：设计预打印的表格

在这一步，外部文档和回转文档被独立出来进行特殊考虑，因为它们包含了大量的固定预打印信息，这些预打印信息必须在设计最终输出之前就设计好。在大多数情况下，预打印表格的设计被交给一个表格生产厂家，但是企业必须说明设计需求，并仔细地检查设计原型。设计需求涉及以下问题：

- 表格中必须显示什么预打印信息？预打印信息包括联系信息、标题、标签和显示在所有拷贝上的其他公共信息。

- 表格应被设计用于邮寄吗？如果要邮寄，并使用开口信封，则地址的位置就很重要。
- 每天要求打印多少表格？每周多少？每月多少？每年又是多少？
- 表格的尺寸有多大？表格尺寸和表格数量可能会影响邮寄费用。
- 表格将被打孔作为回转文档吗？而且，对于回转文档，地址位置变得更为关键，因为外部输出的返回地址成为返回文档的邮寄地址。
- 需要在表格上（正面和反面）打印什么图例、政策和指令？
- 使用什么颜色？用于哪种拷贝？

对于外部文档来说，也可以采用好几种替代方案。复写纸和化学复写纸是最常用的复制技术，可选择的复写纸是一种特殊的复写纸，其中主拷贝中的某个域将不会被打印在一个或多个其余的拷贝中（被省略的域必须同表格生产厂家协商确定）。“双份打印”是一种特殊技术，其中两套表格（可能包括复写纸）在打印机上同时打印。

前面的图 14-3 中显示了音阶公司案例的一个预打印表格。

14.2.3.4 第 4 步：设计、验证并测试输出

设计决策和细节被记录在项目资料库中后，我们就必须设计报告的具体格式，输出的格式或者布局直接影响着系统用户读取和解释输出的能力。布局格式的最佳方法是画草图，或者更好地生成一个例子报告或文档。我们需要向系统用户展示草图或原型，获得他们的反馈意见，并修改例子输出。注意，使用现实的或合理的数据，并演示所有的控制断点十分重要。

在这个设计步骤中，格式最重要。图 14-8 总结了关于打印报告和表报告的一些设计问题和考虑因素，这些设计因素中有许多同样可应用于屏幕输出。而且，屏幕输出还有一些特殊考虑因素，这些因素则在图 14-9 中总结。

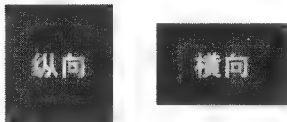
设计内容	设计指南	例子
页面尺寸	曾经有一段时间，大多数报告都在特大型的纸张上打印，这要求使用特殊的装订和存放方式。如今，可供选择的页面尺寸是标准尺寸 $8\frac{1}{2}$ 英寸 \times 11 英寸）和传统尺寸（ $8\frac{1}{2}$ 英寸 \times 14 英寸）。这些尺寸同现代企业中的激光打印机兼容	无
页面方向	页面方向确定页面的长和宽。经常首选的是纵向（例如， $8\frac{1}{2}$ W \times 11 L），因为它与我们绝大多数书和报告的方向一致；但是，对于表格报告来说也经常必须使用横向（例如， 11 W \times $8\frac{1}{2}$ L），因为报告可能要打印很多列	
页面标题	每页都应该有页面标题。页面标题至少应该包括一个可识别的报告标题、日期和时间以及页号，标题可以合并成一行或者使用多行	JAN 4, 2001 第 4/6 页 OVERSUBSCRIPTIONS BY COURSE
报告说明	说明是解释报告中使用的缩写、颜色或代码在一份打印报告中，说明可以只在第 1 页打印，也可以在任何一页都打印 在一个显示屏中，说明可以作为一个弹出式对话框提供	报告说明 SEATS 教室中座位的数量 LIM 课程注册人数上限 REQ 系里申请的座位数 RES 系里预定的座位数 USED 系里使用的座位数 AVL 系里可用的座位数 OVR 系里超额认购的座位数

图 14-8 表报告设计原理

设计内容	设计指南	例子
列标题	列标题应该简短，并具有描述性，如果可能，应该避免使用缩写，不过这一点并不总是能够实现。如果使用了缩写，应该包括一个报告说明（见“报告说明”）	可自我解释
标题对齐	列标题与列下面实际数据的位置关系会极大地影响可读性。对齐方式应该让用户测试是否喜欢，尤其注意是否存在错误解释信息的可能	左对齐（适用于较长的域和可变长度的域） NAME XXXXXXXX X XXXXXXXX XXXXXX 右对齐（适用于一些数字域，特别是货币域），注意要对齐小数点 AMOUNT \$\$\$, \$\$\$. ^{cc} 居中对齐（适用于固定长度的域和一些中等长度的域） STATUS XXXX XXXX
列间距	列之间的间距影响可读性。如果列靠得太紧，用户可能不能正确地区分列；如果它们离得太远，用户又可能很难在页中定位行。一般规则是：每个列之间具有 3~5 个空格	可自我解释
行标题	第 1 列或者头两列应该作为区分每行的标识数据行应该以一种支持其使用方式的形式排序，它经常按照数字顺序或者字符顺序排序	按数字排序： STUDENT ID STUDENT NAME 999-38-8476 MARY ELLEN KUKOW 999-39-5857 按字符排序： SERVICE CANCEL SUBSCR TOTAL HBO 45 345 7665
格式化	数据经常按照无格式字符存储，以节省存储空间。输出则应该重新格式化数据，以匹配用户的规范	存储形式： 输出形式： 307877262 307-87-7262 8004445454 (800) 444-5454 02272000 Feb27, 2000
控制断点	某些行经常表示有意义的数据组，这些数据应该在报告中逻辑地分组。从一个组到下一个组的转换被称为控制断点，而且经常其后面跟着那个组的小计	RANK NAME SALARY CPT JANEWAY, K 175 000 CPT KIRK, J 225 000 CPT PICARD, J 200 000 CPT SISKOW, B 165 000 CAPTAINS TOTAL 765 000—控制断点 LTC CHAKOTAY 110 000 LTC DATA 125 000 LTC RIKER, W 140 000 LTC SPOCK, S 155 000 EXEC OFFCR 530 000 TOTAL
报告结尾	应该清楚地指出报告的结尾，以保证用户获得完整报告	*** END OF REPORT ***

图 14-8 （续）

屏幕设计要素	设计指南
大小	不同的显示器支持不同的分辨率，设计人员应该考虑使用“最低的常用分辨率”默认的窗口大小应该小于或者等于用户分辨率最低的显示器。例如，如果某些用户只有 640 × 480 像素的显示器，那么就不要设计打开后具有 800 × 600 像素的窗口
滚动	联机输出的优势不受实际页面的限制。但是如果重要信息（例如列标题）滚出屏幕，这也会成为一个缺点，所以如果可能，应该把重要标题锁定在屏幕顶部
导航	用户应该总是清楚自己是在一个联机屏幕的网络中，所以用户还需要具有在屏幕之间导航的能力 WINDOWS：输出显示在被称为表单的窗口中，一个表单可以显示一个记录或者多个记录。滚动条应该指示你处于报告中的什么位置。通常还应该提供按钮，以便在报告的记录之间向前和向后移动，以及退出报告 INTERNET：输出显示在称为页面的窗口中，一个页面可以显示一个或者多个记录。可以使用按钮或超链接在记录之间导航，也可以使用定制查询引擎导航到报告中的特定位置
分区	WINDOWS：区域是表单中的表单。每个表单都独立于其他表单，但可以相互关联，区域可以独立地滚动，Microsoft Outlook 就是一个例子。区域可以用来表示报告约定或控制变值，它们把用户带到报告中不同的节 INTERNET：帧是页面中的页面，用户可以在页面内独立地滚动。帧可以以多种方式改进报告，可以用于表示报告约定、目录或总结信息
信息隐藏	联机应用（例如那些在 Windows 上运行或者在因特网浏览器中运行的应用）提供了隐藏信息直到信息变成需要的或者重要的时候再显示的能力。这种信息隐藏的例子有： <ul style="list-style-type: none"> “深入连接（drill-down）控制”显示最小的信息，然后向用户提供简单的方式来扩展或者减小显示信息的详细程度 在 Windows 输出中，使用数据记录左边小框中的加号或减号将记录展开或者将记录合并。所有这些展开和合并都在输出窗口的内部 在内联网应用中，任何总结信息都可以突出显示成一个超链接，以便把那个信息扩展到更详细的程度。通常，展开的信息在一个独立窗口中打开，以便读者可以使用浏览器的向前按钮和向后按钮在不同详细程度的信息之间切换 信息可以触发弹出式对话框
突出显示	报告中可以使用突出显示来引起用户对出错信息、异常数据或特殊问题的注意。如果使用不当，突出显示也会分散用户注意力。有关人的因素的研究将继续指导我们对突出显示的使用。突出显示的例子包括： <ul style="list-style-type: none"> 颜色（不要使用色盲的人不能分辨的颜色） 字体（变换字体可以提起注意力） 对齐（左对齐、右对齐或居中对齐） 连字符（不建议在报告中使用） 闪烁（可以引起注意，也可能会使人反感） 反显
打印	许多用户仍然喜欢打印的报告，所以应该向用户提供打印报告永久拷贝的功能。对于因特网用户来说，报告可能需要按照工业标准格式提供（例如 Adobe 公司的 Acrobat），这使得用户可以使用免费的广泛使用的软件打开和阅读那些报告

图 14-9 屏幕报告设计原理

音阶公司的管理层担心“MEMBER RESPONSE SUMMARY”报告输出可能会太长了，他们经常只对会员响应一个或几个不同促销商品的信息感兴趣。管理人员需要有“定制”输出的能力，所以，用来让管理人员说明定制需求的屏幕，以及包含实际信息的报告和图形，都应该进行原型化。图 14-10a 表示了用户可以用来选择特定报告（或图形）并定制其内容的屏幕。请注意以下要点：

- ① 选项卡对话框供用户选择获得一个报告或者图形，其中选项卡控制用来表示一系列相关信息。如果用户选中了 Graphs 选项卡，就显示用来输出定制成一个图形的信息。
- ② 下拉列表用来选择期望的报告，用户可以单击向下箭头列出可能的报告以进行选择。
- ③ 提供给用户一系列复选框，它们分别对应用于定制所选报告的一般规则，用户可以“选择”那些希望出现在报告中的内容。
- ④ 使用一组复选框让用户选择一个或多个希望包括在报告中的产品类别。

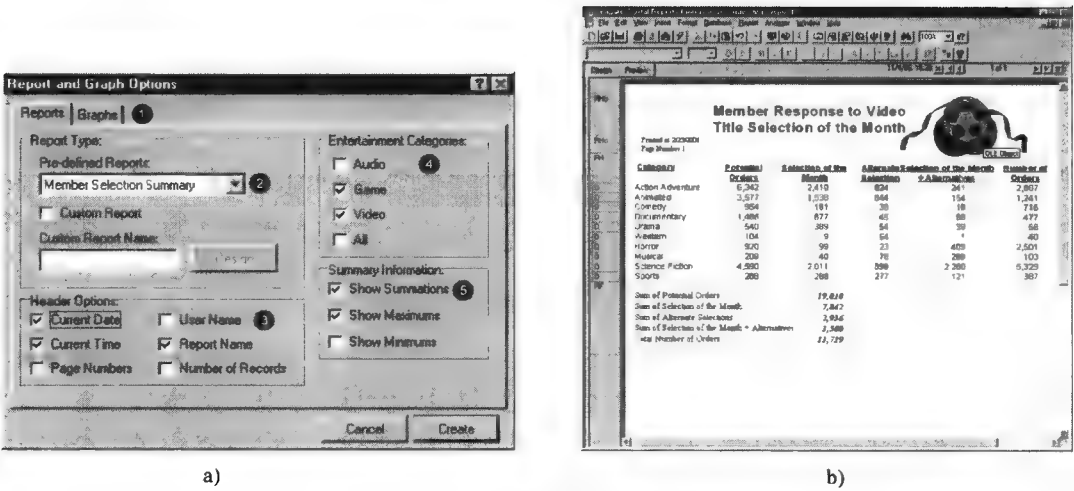


图 14-10 报告定制和表报告原型

a) 报告定制原型 b) 表报告原型

再次使用一组复选框让用户进一步定制报告，并允许用户指出总结信息的类型，或者每个产品类别希望的总数。

现在让我们看一个从前面的报告定制对话框生成的报告原型，图 14-10b 是实际报告的屏幕输出版本。请检查表设计的内容和显示，注意系统允许用户通过垂直和水平地滚动来查看整个报告。另外，系统提供了按钮让用户可以触发向前翻页和向后翻页，以查看不同的报告页面。

最后，让我们看一个“MEMBER RESPONSE SUMMARY”报告输出的图形版本（见图 14-11），请注意以下内容：

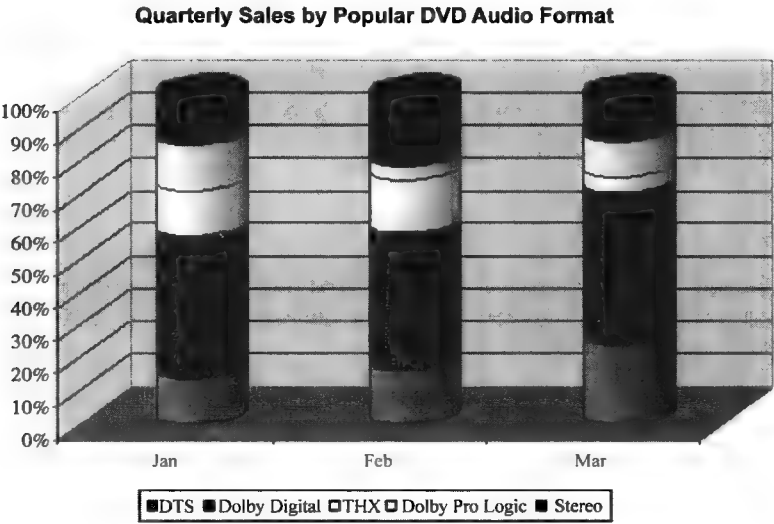


图 14-11 图形报告原型

- 图形沿着垂直轴和水平轴标记。
- 提供一个图例用来辅助解释图形条。

当设计输出原型时，让用户参与进来以获得反馈很重要。应该允许用户实际地使用或测试屏幕，这包括：演示用户如何获得合适的帮助或指示，进一步获得额外的信息，通过页面导航，请求可用的不同格式输出，调整输出大小，以及测试定制能力。总之，所有的特征都应该被演示或测试。

到目前为止，我们仅仅显示了一个表报告和一个图形报告的例子；另一类输出是一次一个记录的

报告, 用户可以向前或向后浏览文件中的单个记录。图 14-12 的例子显示了一个“一次一个记录的报告”。请注意以下内容:

- ① 每个域都被明显地标识出来。
- ② 增加了按钮用于在记录之间导航。几乎被大家广泛接受的按钮有“第一个记录”(FIRST RECORD)、“下一个记录”(NEXT RECORD)、“前一个记录”(PREVIOUS RECORD)和“最后一个记录”(LAST RECORD)。
- ③ 我们为用户增加了三个按钮, 一个用来刷新输出, 一个用来获得输出的一份打印拷贝, 还有一个用来退出程序(程序员以后要为退出编写代码)。

14.2.4 基于 Web 的输出和电子商务

我们想涉及的最后输出设计内容是基于 Web 的输出。音阶公司案例项目将各种电子商务和电子商务功能添加到会员服务信息系统中, 其中一些功能会影响到输出设计。

项目的一个逻辑输出需求是目录浏览, 会员应该能够浏览并查找目录, 而目录本身就是一个输出。图 14-13 是用于实际目录输出的一个原型屏幕, 请注意以下注释:

图 14-12 单个记录输出原型

图 14-13 Web 数据库输出

- ❶ 这个输出使用帧区分导航和输出。
- ❷ 屏幕使用超链接提供复杂菜单结构的导航，这些内容都与输出有关。
- ❸ 用户可以通过超链接得到额外信息，这个功能被称为“深入连接”。
- ❹ 阴影用来区分详细产品信息，这种实现方式反映了设计 Web 输出的更艺术化方式。而且，“BUY”按钮有效地把这个输出转换成到后续输入的一个触发器，这是一个回转文档的电子商务虚拟等价。
- ❺ 大多数 Web 输出屏幕设计要求在屏幕上有标准的脚注，以提供额外导航。
- ❻ 图片可以是可选择的对象。在这种情况下，它表示另一种类型的“深入连接”，通过它用户能够获得额外信息。

另一种输出需求是让会员播放产品的录像带和磁带以预览要购买的产品，这种预览将由前面屏幕的一个超链接触发，并激活一个多媒体播放器，如图 14-14 所示。这种输出扩展有可能成为因特网和内联网应用的规范而得到广泛应用。请注意以下注释：

- ❶ Web 输出经常使用插件，这个输出屏幕具有与一个典型音频或视频播放器关联的标准按钮。
- ❷ Web 输出通常也提供会话需要的相应插件。



图 14-14 Windows Media Player 输出

复习题

1. 原型系统具有什么特征?
2. 输出如何分类?
3. 总结报告和例外报告之间有什么区别?
4. 有哪些外部报告的例子?
5. 表格输出和分区输出之间有什么区别?
6. 为什么打印报告还需要额外的屏幕输出?
7. 举一些图形表格的例子。
8. 为什么应该使用图形化输出?
9. 有哪些输出设计指南?
10. 设计输出有哪些基本步骤?
11. 当说明物理输出需求时, 分析员应该考虑的两类最重要的评价标准是什么? 它们为什么重要?
12. 分析员需要考虑哪些设计问题?
13. 预打印表格的用途是什么?
14. 当在因特网上显示信息时, 使用框架有什么优势?

问题和练习

1. 100 年前, 如果你正在设计一份报告, 可以使用哪些不同的发布方法和介质? 50 年前呢? 如今呢? 在过去的 100 年中, 你认为在报告方面最大的变化是什么?
2. 你是县里的社会服务部的系统设计人员。县儿童保护部门的主任关心该部门的待处理案件数量和案件保持开放的时间长度。部门的目标是没有超过 60 天的正在办理的案件, 最好没有超过 30 天的。主任想要一份月报显示案件的数量, 按照案件的时间和部门的 12 个儿童保护工作人员进行分类。你应该设计哪类报告? 输出格式应该是表格的还是分区的? 描述定义报告的逻辑需求的数据结构。使用本章描述的格式。
3. 使用问题 2 的信息创建一个报告的原型; 使用一个自动化工具例如 Microsoft Access (或者如果你喜欢, 可以用老式的方法创建原型。) 在报告中填上几个例子数据, 按照工作人员姓氏的字母顺序。
4. 儿童保护部门的主任对报告很满意, 但还想看到图形格式的报告。什么类型的表格是不合适的? 为什么? 什么类型的表格对这类报告是合适的? 为什么? 你认为哪种是最好的? 为什么?
5. 你将为汽车代理商销售经理设计哪类报告, 他的工作是检查每周、每年的汽车销售? 你将在报告中包含什么数据元素? 在设计报告前你将会询问销售经理什么问题?
6. 销售经理还想知道以周为单位谁没有完成上周和/或本年度的销售配额。在这种情况下需要什么类型的报告? 你将包含什么数据元素, 如何组合它们?
7. 匹配第 1 列中的定义或例子与第 2 列中的词汇。

A. Web 使能的按需报告归档系统	1. 详细报告
B. ATM	2. 显示输出图
C. 传统的输出介质	3. 外部输出
D. 拖欠债务账号报告	4. 散点图
E. 不同数据组之间的转换	5. 总结报告
F. 库存汽车的报告	6. 控制间隔
G. 网站上的“Buy”按钮	7. 分区输出
H. 很少再使用的屏幕设计工具	8. 回转输出
I. 销售订单	9. 例外报告
J. 显示两个或多个数据序列之间的关系	10. DataWatch Monarch/ES
K. 按区季度销售报告	11. POS 终端
L. 销售收据	12. 微缩胶片

8. 销售经理要你设计一个自动化表格, 按季度显示过去 5 年内的公司年度销售。经理觉得条形图太单调, 而想使用饼图, 在一份容易阅读的报告显示 5 年的销售报告。你如何看待经理的想法? 解释一下。
9. 儿童保护部门的主任对你在问题 2 中设计的总结报告很满意。为了帮助儿童保护工作人员管理他们的案件并排列优先级, 主任现在想让你设计一份给每个工作人员的报告, 显示他们正在进行的案件, 包括每个案件已进行的时间。而且, 报告应该是一份回转文档, 儿童保护工作人员可以提供它们正在进行的案件的状态, 包括估计的完成时间。需要什么类型的报告? 需要什么数据元素? 案件应该按照什么顺序列表? 为这份报告创建一个原型设计。
10. 你是一个大型生产企业的 IT 部门的一名系统设

计人员,该企业拥有遍布全国的工厂。CIO 告诉你主管市场的副总裁想要一份新的主管层面的报告,按地区和办公室显示日产量,以便能够检查生产水平并快速地确定问题。你的 CIO 告诉你在后天要有一个初始的设计和原型系统。基于你所获得的信息,需要哪种类型的报告?它是供内部使用的还是外部使用的?假设公司信息系统已经收集了这份报告所需的数据,还剩下哪些设计问题?

11. 在前面的场景中,如果你的企业没有使用 CASE

项目和研究

1. 在 20 世纪 90 年代(甚至更早),有大量关于无纸化办公的讨论。一些业界的预言家和未来学者预测在不久的将来,在许多企业中纸将成为过去的记忆。然而如今现实似乎十分不同,事实上,商业界正在消耗掉比以前更多的纸张。在网上对于这个主题现在和过去的文章进行一番研究。
 - a. 描述你找到的文章。
 - b. 比较他们的观点。
 - c. 联系你所在地区的一个大型企业和一个政府部门,它们是否考虑将无纸化办公作为目标?如果是,它们计划如何实现,有所进展吗?
 - d. 你自己的企业或学校呢?
 - e. 在这个主题上你自己持什么观点?你认为无纸化办公是可行的概念吗?为什么?
2. 设计一个表单或界面屏幕,用来比较观察一个奥运会体操运动员:它看上去十分简单,直到你实际试着自己做一做。考虑以下的问题:
 - a. 基于你自己的经验,以及你从本书和其他课本的阅读,什么使得一个表单或界面屏幕“好”,而另一个“坏”?
 - b. 挑选一个你觉得特别糟糕的表单或者界面屏幕。描述为什么。
 - c. 重新设计这个表单或屏幕使其成为你觉得“好”。
 - d. 让一些学生或者同事比较和评价这个表单或屏幕“前一个版本”和“后一个版本”。他们对你的“后一个版本”比“前一个版本”打分高多少?
 - e. 如果要被收集的数据本身没有很好的设计,你能做出一个设计良好的表单或屏幕吗?为什么?
 - f. 在今天的地球村中,在一种文化中认为设计良好的表单或屏幕界面会被认为是通用的好吗?文化差异对设计有多大的影响?
3. 尽管说我们生活在一个空前的技术变革时代可能是陈词滥调了,但确实很难真正地理解在很短的时间内发生巨大的变化,以及它们对我们的影响。为了有助于理解这些变化,考虑以下问题:
 - a. 确定在过去 1000 年中发展起来的不同的输出方法,并按照时间线绘制它们。你标示出了多少种输出方法,其中有多少是在最近 50 年里市场上可获得的?最近 25 年呢?最近 10 年呢?
 - b. 你能找到的反馈文档的最早的版本是什么?
 - c. 按照你的研究,微缩胶片在什么时候得到广泛使用?它对私人 and 公共领域的企业有什么影响?
 - d. 屏幕输出呢?PC 监视器什么时候开始广泛使用?它们对私人 and 公共领域的企业有什么影响?
 - e. 在今天使用的所有的输出方法中,你认为哪一种对政府和文化的影响最大?为什么?
4. 许多企业已经实现了公司内联网。但看上去相对少的(至少目前)已经将他们的内联网同桌面工具集成,例如 Microsoft Office、电子邮件和日历,以及雇员使用的特殊的数据输入/输出应用。
 - a. 联系几个本地的私人 and 公共领域的企业。它们当前实现了内联网吗?
 - b. 描述如何使用内联网,以及它们具有什么特点。这些内联网中有与桌面工具和/或雇员使用的应用集成的吗?
 - c. 看看你是否能了解到这几种内联网。它们的界面设计得好吗?如果有,当设计内联网屏幕界面时,需要考虑内联网和互联网应用之间有什么差别?
 - d. 如果你有机会为你的企业设计一个完全集成的内联网,设计中将包括什么特征和功能?
 - e. 为你的内联网创建一个原型设计。
5. 工具或者专门的报告编写工具进行屏幕布局和原型设计,你能使用什么工具?对于一份主管层次的报告,在设计输出时需要应用的最关键的原则是什么?(记住,你在公司的未来可能依赖于了解并能够应用这些原理。)
12. 你已经志愿为你的本地图书馆的网站服务。图书馆计划开发一个联机图书目录,可供图书馆资助从他们家里的计算机通过因特网预订图书。这些资助人中许多是年龄较高的公民。有哪些屏幕设计问题应该考虑?

5. 在如今的全球经济环境中，特别期望按需的信息。这是一个最近的发展，对企业和个人具有重大的影响。考虑以下的问题：
 - a. 在 1800 年，如果一个欧洲的商店发送一份反馈报告给它在纽约的代理，公司预期（至少）需要多长时间收到反馈？
 - b. 在 1900 年呢？
 - c. 在 1950 年呢？
 - d. 在 2005 年呢，最多多长时间？
 - e. 描述你认为在报告速度上的变化对企业和个人最大的影响是什么。
 - f. 在报告速度上的这种极度的变化（预期的）是好还是不好？对于企业呢？对于雇员个人呢？

小型案例

1. 收集一份详细报告、总结报告和例外报告的例子。将这些报告以及报告内容的简单描述提交给你的老师。报告之间有什么相似点吗？不同点？
2. 联机表单可以设置成“下载”表单内容到一份电子邮件中，然后发送到指定的账号。寻找完成这个工作的代码片断，创建一个发送内容到你的电子邮件地址的简单的联机表单。填写并提交你的表单至少一次。转发你收到的包含表单内容的电子邮件给你的老师，附上你的表单的 URL。
3. 信息应该只被输入到信息系统一次。毕竟，信息应该在部门之间实现数字化共享，不必重新输入数据。这是为什么？当系统设计人员将数据直接发送到数据库中的联机表单（例如你在问题 2 中设计的）时，你认为他们会遇到哪类常见的数据格式问题？
4. 找一个真正设计良好的输出的例子（可以是一个表单、报告、电子邮件等）。然后找一个设计较差的例子。在课堂上展示这两个例子。组织一次关于改进差的设计例子的讨论，使用来自设计良好的输出例子的特定的良好属性。将按照你使班级投入的能力和作为一个团队成员工作以改进输出介质的情况进行打分。

团队和个人练习

1. 个人练习：大学、职业和家庭（没有提到我们所做的所有事情）都要花费大量的时间和精力。反省一下你现在的的生活或者未来如何生活。你在平衡生活和工作方面做得如何？你将如何管理它们之间未来的冲突？你生活中什么是优先的？不用提交任何工作。你可以按照你希望的，以圆桌讨论的形式讨论。
2. 个人练习：作为一个有经验的系统人的一部分是阅读和理解别人。注意有些人看上去有点失败或者不幸。为那个人做一些好事。不要介意你是否认识他或她。
3. 对于很快就要毕业和找工作的学生：去一个文具店购买一些精美的信纸给面谈人员写感谢的话。作为一个团队，准备 1) 一封正式的感谢信（给工作面试人员）；2) 一套你认为会在面试中问到的面试问题。在课堂以外，花些时间同朋友或家庭模拟面试。

输入设计和原型化

本章概述和学习目标

本章讲授如何设计计算机输入，本章是用于讨论客户/服务器或者 Web 系统的使用图形用户界面的联机系统设计三章中的第 2 章。本章将介绍以下内容：

- 为计算机输入定义合适的格式和介质。
- 解释数据收集、数据录入和数据输入的区别。
- 确定并描述几种自动数据收集技术。
- 将人的因素应用到计算机输入设计中。
- 为计算机输入设计内部控制。
- 为显示在 GUI 输入屏幕上的输入属性选择正确的屏幕控件。
- 设计一个 Web 输入界面。

输出和输入设计代表了某种类似“鸡和蛋”的问题——你先做哪一个？本书首先介绍输出设计，经典的系统设计称这种方式为系统验证测试——设计输出，然后确定足够的输入以产生输出。在实践中，任务的顺序并不那么重要，因为现代系统分析技术已经充分地预定义了逻辑输入和逻辑输出需求。如果愿意的话，完全可以改变第 14 章和第 15 章的学习顺序。

本章关键术语

数据收集（data capture）是新数据的标识和获取。

源文档（source document）是用来记录业务事务的表格。

数据录入（data entry）是把数据翻译成计算机可读格式的过程。

批处理（batch processing）是一种数据处理方法，其中许多事务的数据被收集到一个文件中进行处理。

联机处理（online processing）是一种数据处理方法，其中事务的数据立即处理。

远程批处理（remote batch processing）是一种数据处理方法，其中数据联机输入，收集成一批，以后再处理。

15.1 输入设计概念和指南

管理人员和用户根据系统输出（见第 14 章）做出重要决策，这些输出要么从输入数据产生，要么从访问数据库得到的数据产生，而且数据库中的任何数据必须首先被输入。在本章中，将学到如何设计计算机输入，输入设计服务于一个重要目标——收集和获取数据，并把它们转换成适合计算机使用的格式。

如今大部分输入通过快速构建原型进行设计，这些原型可以是计算机生成的简单模拟程序，或者是从原型数据库（例如那些为 Access 开发的数据库）产生的程序。原型一般功能不完整，它们没有包括系统最终版本中所需的安全特征、数据编辑或者数据修改功能。进一步来讲，由于考虑到生产效率因素，它们可能没有包括那些应该包含在一个实际运行系统中的每个按钮或控制特征。

在需求分析期间，输入被建模成包含数据属性的数据流。甚至在最全面的需求分析中，我们也有可能遗漏某些需求，因此，输入设计可以将新属性或数据域引入到系统中。如果输出设计将新属性引入到输出中，这一点就更加重要——输入必须总是足够产生输出！

下面首先讨论输入的类型。输入可以按照两个特征进行分类：1）数据最初如何收集、输入和处理；2）用来收集和输入数据的方法和技术。图 15-1 演示了这个分类法，其中的特征将在后续小节中简要讨论。

15.1.1 数据收集、数据录入和数据处理

当考虑“输入”时，你通常想到的是输入设备，例如键盘和鼠标，但是输入过程在数据到达这些设备之前早就开始了。为了实际地将业务数据输入到计算机中，系统分析员可能要设计源文档、输入屏幕和用于使数据进入计算机的方法和程序（从客户到表格到数据录入职员到计算机）。

过程 方法	数据 收 集	数 据 录 入	数 据 处 理
键盘	数据通常通过一个业务表格收集，业务表格成为输入的源文档 可以通过电话实时收集数据	数据通过键盘录入。这是最常用的输入方法，也是最容易出错的输入方法	老式：数据被收集到批文件（磁盘）中作为一批进行处理 新式：只要数据一被录入就立即进行处理
鼠标	同上	同键盘一起使用，简化数据录入 鼠标作为屏幕的单击设备，可以用在图形用户界面中通过单击选择减少错误	同上。但鼠标的使用通常与联机处理和实时处理有关
触摸屏	同上	数据在触摸屏或者手持设备上输入 数据录入人员要么触摸命令和数据选择，要么使用手写识别输入数据	在 PC 上，触摸屏的处理同上 在掌上电脑上，数据存储在掌上电脑中，供以后作为远程批处理进行处理
销售点	数据收集尽可能离销售点近，不使用源文档	数据经常直接由客户输入（例如 ATM），或者由一个直接同客户交互的雇员输入（例如收银机）输入需要使用特殊的专用终端，并利用本表中其他一些技术	数据几乎总是随着事务或查询一起处理
声音	数据收集尽可能靠近数据源，即使客户是位于远程（例如，在家里或者在上班的地方）	数据使用按键音（一般来自电话）输入，通常需要相当严格的命令菜单和有限的输入选项	数据几乎总是随着事务或查询一起处理
语音	同声音	数据（和命令）是说出来的。这项技术还不像其他技术那样成熟和通用	数据几乎总是随着事务或查询一起处理
光标记	数据以标记或者精确格式化的字符、数字和穿孔形式记录在光扫描纸上，这是一种最古老的自动数据收集形式	消除了数据录入的需要（常用于教育界，例如测试评分、课程评价和考试）	数据几乎总是作为一批进行处理
磁性墨水	数据通常事先记录在以后要由客户填写的表格上 客户在表格上记录其他数据	磁性墨水阅读器读取磁化的数据 客户添加的数据必须使用其他方法录入 这项技术用在需要高度正确性和安全性的应用中，最常用的是银行支票（支票号、账号、银行 ID）	数据几乎总是作为一批进行处理
电磁传送	数据直接记录在数据描述的对象上	数据通过无线电传输	数据几乎总是立即处理
智能卡	数据直接记录在数据描述的客户、雇员或其他人携带的设备上	数据通过智能卡阅读器读出	数据几乎总是立即处理
生物识别	唯一的人类特征成为了数据	数据由生物传感器读取，主要应用是安全和医疗监控	数据立即处理

图 15-1 一个输入分类法

这把我们带到了第一个基本问题：数据收集和数据录入的区别是什么？数据发生了！它伴随着称为事务的业务事件发生，例如 ORDER（订单）、TIME CARD（计时卡片）、RESERVATION（预约）等。当“数据发生”时，我们必须确定何时以及如何收集数据。

数据收集是新数据的标识和获取。什么时候容易收集数据？数据一出现就尽可能快地收集总是最好的。如何收集就是另一回事。过去，人们使用称为源文档的专门的书面表格来收集数据。源文档是用来记录业务事务的表格，用以描述事务数据。

可以复制几乎任何书面表格的显示屏幕逐渐取代了书面表格，这种趋势由于基于 Web 的电子商务和电子业务的发展而加快步伐，但业务表格仍然常常用作源文档供数据录入之用。源文档的设计要仔细，其布局和可读性将影响数据录入的速度。

数据录入是把源数据和文档翻译成计算机可读格式的过程。因为数据录入曾经完全基于键盘，所以企业雇用了一批数据录入职员。随着联机计算变得越来越普及，数据录入的责任直接转移到了系统用户。如今另一种转移正在出现，由于个人计算机和因特网的发展，一些数据录入已经直接转移到了客户。无论哪种情况，数据录入都将产生用于数据处理的输入。

输入的数据必须后续地被处理——数据处理。在本章中，我们不关心数据如何转换成输出，但对输入处理的时效性感兴趣，即输入数据什么时候被处理？

15.1.1.1 批处理

批处理曾经是数据处理的主要形式。在**批处理**中，输入的数据被收集到称为批量数据的文件中，每个文件作为许多事务的一批处理。与常见的看法不同，有些数据仍将成批处理，计时卡片就是经典的例子。大多数批次以磁盘文件记录（所以有词汇 key-to-disk），但有些较老的系统可能仍在磁带中记录批次（key-to-tape）。

15.1.1.2 联机处理

如今大多数（但不是所有的）信息系统已经被转换成联机处理系统。在**联机处理**中，收集的数据立即处理。最初，数据在终端输入。如今，同样的数据在 PC 和工作站上收集，以发挥它们在把数据发送到服务器计算机之前进行数据验证和编辑的能力。由于 PC 如此普及，很少再听到联机处理这个词了，我们通常听到的是客户/服务器，其中 PC 是客户端。

如今绝大多数应用系统表现给用户一个基于 PC 的图形用户界面（GUI）。Windows 是现今企业中的主流 GUI，但是作为因特网和内联网应用平台的 Web 的兴起可能使得 Web 浏览器成为未来最重要的用户界面，Internet Explorer 和 Firefox 是如今市场上的主流浏览器界面。本章将既涉及 Windows 客户/服务器界面的输入设计，也涉及浏览器界面的输入设计。

15.1.1.3 远程批处理

批处理和联机处理代表了处理图谱的两个极端，实际上还存在一个组合方案——远程批处理。在**远程批处理**中，使用联机编辑技术输入数据；但是，数据被收集成批次而不是立即处理。以后，再进行批处理。

现代的远程批处理可能有几种形式。一个简单的例子是使用基于 PC 的前端应用系统收集和存储数据，数据以后可以通过网络传输到远端进行批处理。远程批处理的一个更现代的例子是使用断开连接的笔记本电脑或掌上电脑（或设备）收集数据供以后处理。如果你最近从 UPS 公司或者联邦快递公司收到过包裹，你就应该看到司机使用这种设备记录签收和发货。

现在，已经讨论了基本的数据收集、数据录入和数据处理技术，下面可以更详细地介绍图 15-1 中的各种输入方法。

15.1.2 输入方法和实现

不同的输入设备（例如键盘和鼠标）在绝大多数介绍性的信息系统课程中都有所讲述。在本节中，重点介绍输入方法及其实现，而不是输入技术。就像上一节描述的那样，我们对方法的选择如何影响数据收集、数据录入和数据处理特别感兴趣。当介绍这些方法时，应该继续参考图 15-1。

15.1.2.1 键盘

键盘数据录入仍是最常见的输入形式。不过键盘数据录入需要大量的数据编辑工作，因为人们从源文档键入数据时难免会出错。现在，图形用户界面（例如 Windows 和 Web 浏览器）使得设计联机屏幕成为了可能，这种界面强制用户做出正确的选择，从而减少了错误。将在下一节中介绍这类界面中几个有用的 GUI 控件。

15.1.2.2 鼠标

鼠标是一种用在图形用户界面中的单击设备，它有助于导航联机表格、单击命令和输入选项。例如，一个属性的合法值可以在屏幕上显示成“可点选的”方框或按钮，这省去了键入那个数据的必要，从而减少了数据录入错误。将在本章的输入设计中介绍基于鼠标的控制。

15.1.2.3 触摸屏

触摸屏显示器是一种新兴的输入技术，这种技术在不久的将来会极大地影响输入设计。这类显示器在手持设备和掌上电脑（见右图）中很常见，它们正应用于无数的信息系统应用中。这类设备简化了仓库和生产车间的许多数据收集活动，你可以通过对触摸屏按钮编程来收集数据。大多数这类设备也支持手写识别，图中显示的 Symbol Technologies 设备还能扫描和阅读条形码（这里只是简短地介绍了该设备）。



掌上电脑

15.1.2.4 销售点

销售点（POS）终端已经使用了一段时间了，它们取代了旧式的收银机。这些终端在销售点收集数据，并提供了节省时间的数据输入方式进行事务计算，还能产生一些输出。类似于手持设备，大多数销售点终端可以扫描和阅读条形码以消除键入错误。自动柜员机（ATM）（另一种形式的 POS 终端）则直接由客户操作。

15.1.2.5 声音和语音

声音代表了另一种形式的输入。你可能已经使用过基于按键式电话的系统来注册课程。这种基于语音的系统要求使用特殊的输入/输出设计技术，这些系统和技术超出了本书的介绍范围。

这种输入方法的一种更复杂形式是使用语音识别技术输入数据。目前这项技术还不太成熟，也不太可靠，所以最好用它来输入命令，而非数据。但是语音识别技术代替键盘作为主要数据输入方法总有一天会实现。

其他输入方法基本上可以归类为自动数据收集（Automatic Data Capture, ADC）。随着目前输入技术的进步，我们可以消除输入过程中的大部分（有时全部）人为干涉。通过消除这些人为干涉，可以减少与之相关的时间延迟和错误。

15.1.2.6 光标记

用于输入的光标记识别（Optical Mark Recognition, OMR）技术已经存在几十年了，它主要是面向批处理的，典型的例子是用于基于目标问题（例如多项选择）的光标记表格。这项技术也可用于调查表或任何其他应用中，其中可选数据值的数量相对有限，并且高度结构化。能够从这种输入方法获益的大多数应用系统可能已经在使用它了。

光字符识别（Optical Character Recognition, OCR）不太常用，尽管它已经发展成熟。它要求用户或客户仔细地在业务表格上手写输入数据。如果字符和数字书写正确，OCR 阅读器可以不需要人为干涉就能处理表格。显然，成功与否还取决于用户或客户的笔迹，但这项技术确实可行。Columbia House Record Club 公司曾经使用 OCR 表格来让客户响应订单。像大多数 OCR 应用一样，输入域的数量很小（减少了出错的可能性），但对于任何由于不可辨认而被驳回的数据，系统必须实现相应的处理方法。

如今最常用的光技术是条形码。几乎在我们购买的每个产品上都有条形码，但是条形码技术并不限于零售业中。几乎可以为任何企业应用创建条形码，甚至可以在基于 Windows 的应用系统中集成条形码技术，如图 15-2 所示。

15.1.2.7 磁性墨水

磁性墨水 ADC 技术是你可能会遇到的一种技术，通常包括了磁条卡技术，但还可能包括磁性墨水字符识别（MICR）技术。如今有超过 10 亿张磁卡在使用中！它们已经在许多企业应用中找到了出路，例如信用卡交易、构建安全访问控制、雇员签到跟踪。MICR 广泛地应用于银行业。

15.1.2.8 电磁传送

电磁传送 ADC 技术使用无线电频率识别物理对象。这项技术把一个标签和天线附到被跟踪的物理对象上。标签包含了存储器，用来识别被跟踪的对象。只要对象在阅读器产生的电磁场范围内，标签就可以被阅读器阅读。在涉及跟踪视线以外和移动物理对象的应用系统中，这种识别技术越来越常见。例如，电磁 ADC 用于公共运输跟踪和控制，跟踪生产的产品和跟踪动物等。

15.1.2.9 智能卡

智能卡技术有能力存储大量信息。智能卡类似于信用卡，但比它稍微薄一点，它们的不同点在于智能卡包含了一个微处理器、存储电路和一个电池。可以把智能卡看作是一个板子上有计算机的信用卡。它们是一种便携式存储介质，从中可以获得输入数据。虽然这种技术刚开始传入美国，但 60% 以上的法国人每天都在使用智能卡。智能卡应用在健康记录领域特别被看好，其中一个人的血型、接种疫苗以及医疗历史记录都可以通过智能卡获得。其他应用可能有：护照、销售点交易的财务信息、付费电视等。另一项未来的应用可能是一种组合借记卡，它能自动地维护并显示你的账号余额。右图显示了一张用于安全应用的智能卡。

15.1.2.10 生物识别

生物识别 ADC 技术的根据是唯一的人类特征或特质。例如，每个人都可以通过其唯一的指纹、语音模式或某种特征（视网膜或腕关节）模式辨识。生物识别 ADC 系统由收集个人特征或特质的传感器构成，它首先数字化图像模式，然后比较输入图像和存储模式进行辨识。生物识别 ADC 得到越来越多的应用，因为它提供了最准确、最可靠的辨识方式。这项技术在需要安全访问的系统中特别常用。

15.1.3 输入设计的系统用户问题

因为输入源自系统用户，所以人的因素在输入设计中扮演了重要角色。输入应该尽可能地简单，并且应该降低输入错误的可能性。必须考虑系统用户的需求，因此在设计输入时，有几个方面的人的因素应该被评估。

应该尽量减少输入的数据量。输入的数据越多，输入的错误可能就越多，输入数据的时间也就越长。所以，应该仔细考虑为输入收集的数据。输入设计中应该遵循以下一般原理：

- 只收集变化的数据。不要输入固定不变的数据。例如，当决定销售订单（SALES ORDER）输入中要包含什么元素时，对于订购的所有部件，都需要部件编号（PART NUMBER）。但是，需要输入部件描述（PART DESCRIPTION）吗？部件描述可能存储在一个数据库表中，只要我们输入部件编号，就可以查询到部件描述。永久的（或者半永久的）数据应该存储在数据库中。当然，为维护数据库表，我们还必须设计相应的输入。
- 不要收集可以在计算机程序中计算和存储的数据。例如，如果输入订购数量（QUANTITY



图 15-2 Windows 应用系统中的条形码



一张智能卡

ORDERED) 和单价 (PRICE), 就不需要输入总价 (EXTENDED PRICE), 因为它等于订购数量 \times 单价。另一个例子是在表 (数组) 中合并联邦税金扣款 (FEDERAL TAX WITHHOLDING) 数据, 而不是每次都键入那个数据。

- 使用相应属性的编码。编码在前面已经介绍过, 在计算机程序中则可以通过使用表翻译编码。

如果使用源文档收集数据, 对系统用户来说, 源文档应该容易填写并容易录入到系统中。下面的建议可能会有所帮助:

- 包含填写表格的指示。记住人们不喜欢阅读打印在表格背面的指示。
- 尽量减少手写的工作量。许多人书法很差, 数据录入人员或 CRT 操作员可能会误读数据并输入错误数据。尽可能使用复选框, 以便系统用户只需要选中合适的值就可以了。
- 输入 (键入) 的数据应该被排序, 以便它们可以像本书一样 (自顶向下和从左到右) 地阅读。图 15-3a 演示了一个好的流程。系统用户不应该为了输入数据而在一行上从右到左移动, 或者在表格上来回跳转, 如图 15-3b 所示。

a)

b)

图 15-3 好的流程和差的流程

a) 好的流程 b) 差的流程

- 尽可能使用具有已知含义的设计。这方面的典型例子是个人财务应用软件 Quicken, 这个程序的易用性体现在其屏幕上显示的支票簿, 而且支票填写看上去确实像它的书面等价物。不是所有的输入都需要采用模仿方式, 但是一定的模仿可能会很有用处 (见图 15-4)。
- 关于 GUI 屏幕设计的数据输入还有几个其他的指南和问题, 在本章后面讨论用于输入设计的 GUI 控件时, 以及在输出设计和用户界面设计章节中, 我们会相应地介绍这些指南。

15.1.4 内部控制——输入数据的编辑

内部控制在所有计算机系统中都是必要的。内部输入控制确保输入到计算机的数据是正确的, 并且确保系统不会由于意外的和故意的原因而产生错误和被滥用 (包括欺骗)。内部控制指南如下:

1. 应该监视输入的数量。这对于批处理输入特别重要, 因为有些文档可能放错了位置、丢失了或者被跳过了。

- 在批处理系统中, 关于每一批次的统计数据都应该记录在一个批次控制片上。这些数据包括“批次编号”、“文档数量”和“控制总数” (例如, 文档中行条目的数量)。处理完成后, 这些总量可以同报告中的输出总量进行比较。如果总量不相等, 就必须找出原因。

图 15-4 模仿方式的屏幕设计

- 在批处理系统中，一种替代的控制方法是一对一的检查。每个源文档将匹配对应的历史报告细节行，证实文档已经被处理。这种控制检查可能仅仅当批控制总量不匹配时才需要使用。
- 在联机处理系统中，每个输入事务应该登记在一个独立的审计文件中，以便如果存在处理错误或者数据丢失，可以恢复和重新处理数据。

2. 必须确保数据是有效的。有两类错误可能会渗透到数据中：数据录入错误和系统用户记录的无效数据。数据录入错误包括拷贝错误、错位（把 132 键入成了 123）和滑位（把 345.36 键入成了 3453.6）。下面的技术可以用于验证数据：

- 存在性检查 确定输入中所有要求的域是否都实际上被输入了，要求的域应该明显地标识在输入屏幕上。
- 数据类型检查 确保输入了正确的数据类型，例如，字符数据不允许出现在数字域中。
- 区域检查 确定是否每个域输入的数据都位于域定义的合法取值范围内。例如，工资（PAY RATE）可能设置一个上限范围，以确保雇员工资不会比上限还高。
- 组合检查 确定两个域之间的一个已知关系是否有效。例如，如果汽车生产厂家（VEHICLE MAKE）是 Pontiac，那么车型（VEHICLE MODEL）就必须是由 Pontiac 生产的轿车构成的有限值集合（Firebird、Grand Prix 和 Bonneville 等）。
- 自我校验数字 确定同主键相关的数据录入错误。校验数字是添加到主键域中的数字或者字符，它通过应用一个公式（例如 Modulus 11）计算得到实际的键。校验数字可以通过两种方式验证数据录入的正确性：当数据由系统用户输入时，有些数据录入设备可以通过应用相同的公式到数据上自动验证数据。如果输入的校验数字与计算出的校验数字不匹配，就显示错误；另外，计算机程序也可以通过使用准备好的子路线验证校验数字。
- 格式检查 按照数据的已知格式需求对比输入的数据，例如：有些域可能要求以零开头，而另一些则不要求；有些域使用标准标点符号（例如，社会保险号或者电话号码）。值“A4898 DH”可以通过格式检查，而类似的值“A489 ID8”则不能。

在第 13 章中，了解到大多数数据库管理系统执行类似上面描述的数据验证检查，那么为什么还需要输入控制呢？很简单，如今大多数应用都是网络应用，有错误的数据既是一个网络流量瓶颈，也会影响事务吞吐量和响应时间。尽可能接近源头收集和改正输入错误总是最好的，因此我们强调输入控制和验证。

15.2 输入设计的 GUI 控件

如前所述，如今正在开发的绝大多数新应用系统都具有图形用户界面（GUI）。图形用户界面大部分基于 Windows，但是组合了基于 Web 的电子商务的因特网技术的普遍采用迅速地促使一些界面转向采用 Web 浏览器。虽然 GUI 设计提供了更加用户友好的界面，但是它们也带来了比其前任更复杂的设计问题。本章不会去讨论所有的 GUI 设计问题，关于这个主题已经写过很多本书，比较受欢迎的几本书在后面“推荐读物”中列出。

相反，本章将集中介绍如何为 GUI 屏幕上的输入数据选择正确的屏幕控件。可以把控件看成是构建用户界面的“窗口小部件”，它们被包括在大多数现代的开发环境中，例如微软公司的 Access 和 Visual Studio .NET、Sybase 公司的 PowerBuilder、Inprise 公司的 JBuilder、Symantec 公司的 Visual Café、IBM 公司的 Visual Age 等。这些工具中有许多通过资料库共享 GUI 控件（和代码），这个方法被称为基于资料库的编程。

图 15-5 演示了对包含输入控件和代码的资料库的访问。这种方法基于面向对象和面向组件的编程技术，并且已经普遍用于应用开发中。该图描绘了可以由各种系统分析员或程序员用来设计界面的 GUI 控件。开发人员可以在一个位置定义一个可复用域的大部分属性和约束以及那个域的数据验证代码，一旦完成定义，组织中其他系统分析员和程序员都可以使用这些对象或控件。这种基于资料库的方式保证了域的每个实例都按照一致的方式使用。而且，如果业务规则提出要求，就可以改变资料库条目，而不需要对应应用系统进行额外修改。

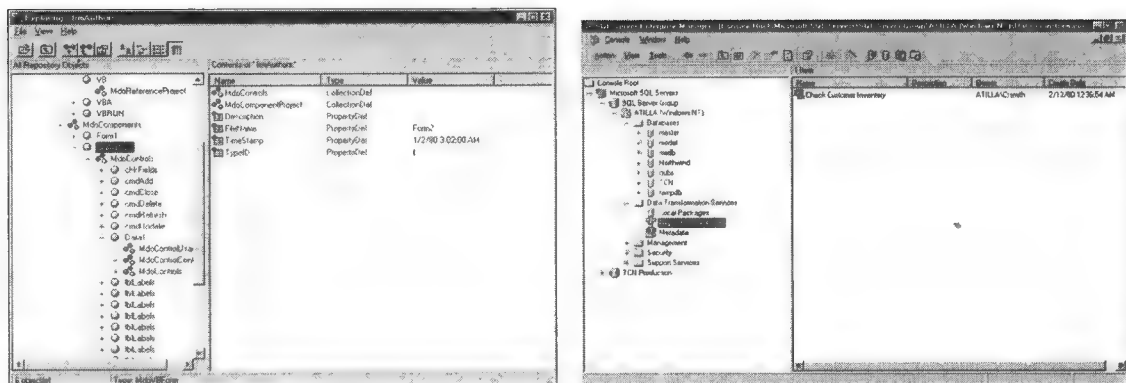


图 15-5 基于资料库的原型化和开发

15.2.1 常用 GUI 输入控件

本节介绍基于 GUI 的输入表格中一些最常用的 GUI 控件，我们将讨论每个 GUI 控件的用途、优点、缺点和使用指南。如果理解了这些内容，就能做出更好的决策，决定屏幕上输入的每个数据属性应该使用哪个控件。我们将把屏幕设计之间的转换推迟到第 16 章介绍。

图 15-6 显示了一个用于输入数据的最常用的屏幕控件库。我们将讨论其中的每个 GUI 控件，它们既可以用于 Windows 界面，也可以用于 Web 界面。

15.2.1.1 文本框①

也许最常用的数据输入 GUI 控件是文本框。文本框通常由一个带有标题的矩形框构成，这个 GUI 控件要求用户在框中键入数据，可以输入单行或者多行数据字符。当文本框包含了多行数据时，通常还具有滚动特征。

当输入数据的值在范围上没有限制，而且分析员不能给用户提供一个有意义的值列表供选择时，使用文本框最适合。例如，单行文本框适合用于收集新客户的名字，因为客户名字的可能值几乎不可能被预先确定。文本框也适合用于收集描述客户提供的订单发货指令数据。同样，发货指令的可能取值几乎是无限的，而且多行文本框更合适，因为发货指令的长度不可预测。在文本框的大小不足以显示整个输入数据值的情况下，文本框可以使用滚动和自动换行特征。

当在输入屏幕上使用文本框时，需要遵循许多使用指南。例如，文本框应该具有一个有意义的描述性标题；标题不要使用缩写；只有标题文本的第一个字母应该大写。

标题的位置也很重要，用户应该能够明显地将标题和文本框联系起来。所以，标题应该放置在实际文本框的左边，或者以左对齐的方式显示在文本框的上面。最后，一般认为标题应该后跟一个冒号以帮助用户区分标题和文本框。

一般来说，文本框的大小应该足够大，以使用户能够输入和查看固定长度输入数据的所有字符。当输入数据的长度可变并且数据长度可能很长时，应该使用文本框的滚动和自动换行特征。

15.2.1.2 单选按钮②

单选按钮为用户提供了一种方便的方法，可以快速地从一值集合中标识和选择一个特定值。单选按钮由一个小圆圈和一个相关的文本描述构成，文本描述对应了相应的选择值，圆圈位于选择值的文本描述的左边。单选按钮通常成组出现——每个值一个单选按钮。当用户从值集合中选择了合适的选项时，对应选项的圆圈就被部分地填充以指示它被选中。当一个选项被选择时，就取消任何默认的

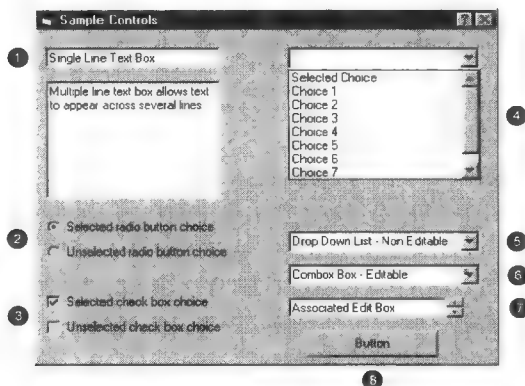


图 15-6 常用 GUI 输入控件

或者以前选中的选项圆圈的选定。单选按钮还为用户提供了通过键盘或者鼠标选择的灵活性。

当期望用户输入具有有限的、预定义的、互斥的值集合数据时，单选按钮最合适。例如，可能要求用户输入订单类型（ORDER TYPE）和性别（GENDER），这两个输入项都具有有限的、预定义的、互斥的值集合。例如，当用户要输入订单类型时，系统可能希望他们从值集合“常规定单”、“紧急订单”或者“固定订单”中选出唯一的一个值。对于性别，系统将希望他们从值集合“女性”、“男性”或者“未知”中选择唯一的一个值。

当使用单选按钮作为数据输入方式时，需要考虑几个使用指南。首先，为了有助于用户浏览，单选按钮应该垂直对齐并以左对齐的方式显示各个选项。如果需要，选项可以以水平对齐的方式显示，但是应该留出足够空间，以方便区分选项。而且，选项组应该可视地组合在一起，以便于把它们同屏幕上显示的其他输入控件区分开。选项组还应该具有合适的有意义的标题。例如，用于“男性”、“女性”和“未知”的单选按钮可以垂直对齐并以左对齐的方式显示，然后在选项组上面以左对齐的方式显示题头/标题“性别”。

对选项的顺序也应该给予考虑。选项数量越多，越应该考虑浏览和辨识选项的方便性。例如：在有些情况下，对用户来说按字母顺序定位选项可能更自然些；而在另一些情况下，值被选中的频率可能对考虑其在选项集合中的定位更重要。

最后，不推荐对于值只是 Yes/No（或者 On/Off 状态）的输入数据项使用单选按钮，这种情况应该考虑使用复选框。

15.2.1.3 复选框③

同文本框和单选按钮一样，复选框也包括两个部分。它由一个正方形框后跟一个输入域的描述文本构成，用户为这个域提供 Yes/No 值。用户既可以使用键盘也可以使用鼠标选择复选框。值为 Yes 的输入数据域用一个填充了“√”的正方形表示，没有“√”意味着输入域的值 No。用户只需将输入域的值从一个值/状态翻转到另一个期望的值/状态即可。

用户经常需要输入值集合由简单的 Yes 或 No 值构成的数据域。例如，用户可能被要求为以下数据输入 Yes/No 值：CREDIT APPROVED? SENIOR CITIZEN? HAVE YOU EVER BEEN CONVICTED OF FRAUD? MAY WE CONTACT YOUR PREVIOUS EMPLOYER? 这些情况都可以使用复选框。复选框控件为用户输入这类数据提供了一种直观可视的方式。

前面的例子是一个简化的场景，其中系统使用独立的复选框。在一个输入屏幕上，系统可能希望用户为一些具有 Yes/No 值的相关输入域输入值。例如，一个医疗门诊的接待员可能要从一个填写好的病人信息表格中输入数据。在该表格的某一部分，病人被问及一些有关疾病的问题，他们可能被问及过去的病史，并被要求从一个各种疾病列表中“选出所有得过的病”。如果设计合理，接待员的输入屏幕将使用复选框控件把每个疾病显示成一个独立的输入域。这些 GUI 控件将在屏幕上实际地关联成一组，而且还要给复选框组提供一个合适的题头/标题。不过要注意，即使复选框可以可视地组合在屏幕上，但是每个复选框仍可以作为一个独立的输入域操作。

遵循建议的这些使用指南将改进复选框控件的使用。注意确保文本描述对用户是有意义的。尽量寻找机会对相关的 Yes/No 输入域组合复选框，并提供一个描述性组标题。

为了帮助用户浏览并从一组复选框中选择，应该垂直对齐并以左对齐的方式排列复选框控件组。如果需要，可以水平对齐它们并确保留出足够空间，这样可以可视化地分隔各个复选框。最后，按照其文本描述合适地排序输入域以给用户进一步的帮助。在大多数情况下，复选框控件的数量很大，所以应该按字母顺序排序；在文本描述说明的是美元范围或者其他度量衡的情况下，可以按照数字顺序排序；在其他情况下，例如把很少几个复选框组合在一起，则可以按照给定输入域的 Yes/No 值被选中的频率排序（所有的输入数据域都使用具有默认值的复选框表示）。

15.2.1.4 列表框④

列表框是要求用户从一列可能选项中选择一个数据项的控件。列表框是一个矩形，其中包含了一行或多行可能的数据值。列表框中的数据值可以采用文本描述的形式，也可以采用图形表示的形式。

具有大量可能值的列表框可以包括滚动条，以便在选项行之间导航。

列表框行包含多列也很常见。例如，一个列表框可能只包括一列，显示称为工作代码（JOB CODE）的数据项。但是，系统可能对它有更高的要求，期望用户识别每个工作代码表示的内容。在这种情况下，为了使工作代码的值有意义，列表框可以包含第2列，也就是包含每个工作代码对应的工作名称（JOB TITLE）。

如何在单选按钮和列表框之间做出选择呢？二者都用于确保用户为数据项输入了正确的数据。当希望选择值固定地对用户可见时，二者也都适用。

决策通常取决于数据项可能值的数量和 GUI 控件可用屏幕空间的大小。滚动能力使得列表框适用于那些屏幕空间有限而输入数据项具有大量预定义的互斥值的情况。

当使用列表框作为数据输入手段时，需要考虑几个使用指南：列表框应具有一个描述性标题；标题不要使用缩写，而且只有标题文本的第一个字母应该大写；标题后跟一个冒号以帮助用户区分标题和方框。

标题的位置也很重要，用户应该能够明显地将标题和列表框联系起来，所以标题应该出现在实际列表框上面左对齐的位置。

还有几个关于列表框的使用指南：第一，建议列表框包含一个突出显示的默认值。第二，注意列表框的大小。通常，列表框的宽度应该足够输入和查看最长的固定长度输入数据项；列表框的长度至少应该显示3个选项，但不超过7个选项；在宽度和长度方面，都可以使用滚动特征向用户表示还有其他选项。

如果使用图形表示，应该确保图形是有意义的，并且真正地代表了那个选项。如果使用文本描述，应该使用大小写混合字母，并确保文本描述有意义。把这些决策或判断建立在用户的角度和观点上很重要。

还应该仔细考虑用户能否方便地浏览和辨识列表框中的选项。为了便于浏览，选项列表应该左对齐。当讨论选项出现在列表中的顺序时，注意要让用户参与。有些情况下，对用户来说选项列表以字母顺序出现是自然的；而另一些情况下，值被选中的频率对考虑其在列表中的位置可能很重要。

15.2.1.5 下拉式列表⑤

下拉式列表是另一种要求用户从一系列可能选项中选择一个数据项的 GUI 控件。下拉式列表由一个带有连接到边上小按钮的矩形选择域构成，小按钮包含了向下箭头和条形图像，这个按钮向用户暗示存在一个隐藏的数据项可能值列表。

当单击按钮时，隐藏的列表就显示出来，并“下拉”到选择域的下面，显示的列表具有同上面提到的列表框控件类似的特征。当用户从选项列表中选择一个值时，被选择的值就显示在选择域中，并且选项列表再一次隐藏起来。

下拉式列表应该用在数据项具有大量预定义值，但是可用屏幕空间不允许使用列表框的情况下。相对前面提到的 GUI 控件来说，下拉式列表的一个缺点是要求用户额外的操作步骤。

用于列表框的许多使用指南可以直接应用于下拉式列表。一个例外是标题的位置，下拉式列表的标题一般以左对齐的方式放置在紧挨着下拉式列表控件选择域部分的上面，或者放置在下拉式列表控件的左边。

15.2.1.6 组合框⑥

组合框组合文本框和列表框的功能，并为用户提供通过文本框输入数据项的值或者通过列表框从一个列表中选择值的功能的选择。

初看起来，组合框十分像一个下拉式列表。但是，同下拉式列表不同的是，矩形框可以作为用户直接输入数据项值的录入域。一旦小按钮被选中，隐藏的列表就显示出来，列表显示在矩形输入域的下面。

当用户从选项列表中选中某个值时，被选择的值就显示在输入域中，并且选项列表再一次对用户隐藏起来。

组合框最适合用于下面这种情况：可用屏幕空间有限，而且期望用户从一个列表中选择一个值，

或者输入一个可能出现在（也可能没有出现在）列表中的选项。

用于下拉式列表的使用指南可以直接应用于组合框。

15.2.1.7 滚动框⑦

滚动框是一个屏幕控件，它包括一个紧跟着两个小按钮的单行文本框。两个小按钮垂直对齐，上面的按钮有一个向上的箭头，而下面的按钮有一个向下的箭头。用户可以使用这个 GUI 控件直接在关联的文本框中输入数据，或者通过鼠标用这两个按钮滚动一系列值进行选择。按钮都具有相关的步进单位。当用户单击箭头按钮时，某个值将出现在文本框中。文本框中的值通过单击箭头按钮处理，向上箭头的按钮将使文本框中的值增加一个步进单位，而向下箭头的按钮将使文本框中的值减小一个相同的步进单位。

滚动框最适合用于让用户通过使用按钮浏览一小组有意义的选择值来做出输入选择，或者直接键入数据值到文本框中。滚动框的数据值应该能够以预测的方式排序。

滚动框应该包含一个明显地标识出输入数据项的标签或者标题，这个标签应该放置在文本框的左边，或者以左对齐的方式放置在紧挨着滚动框的文本框部分的上边。最后，滚动框应该总是在控件的文本框部分包含一个默认值。

15.2.1.8 按钮⑧

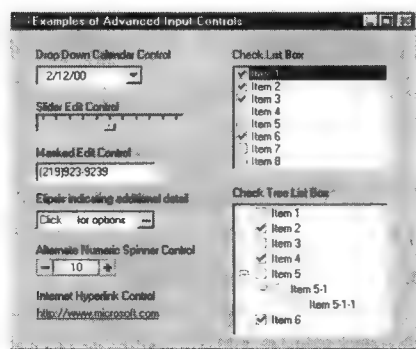
严格地说，按钮不是输入控件，它们没有做出实际数据选择或输入。但无论如何，没有它们，输入表单设计是不完整的。按钮有几个作用：用户可以使用按钮提交所有数据进行处理；或者取消一个事务；或者获得帮助。按钮也可以用来在相同的表单实例之间导航。

还有更多的屏幕控制可用于设计图形用户界面，上面介绍的是用于收集输入数据的最常用 GUI 控件。还有其他 GUI 控件可用，而且你应该熟悉它们以及它们在输入数据中的正确用法。在后面的章节中，你还会接触到几个用于其他目的的 GUI 控件。时刻注意 GUI 领域的发展，新的 GUI 控件会不断出现。

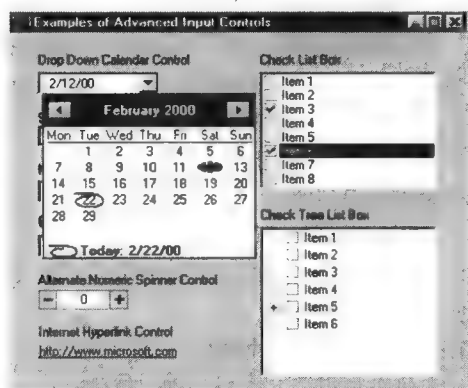
15.2.2 高级输入控件

图 15-7a 和 b 演示了其他一些数据输入控件，这些高级 GUI 控件可以用于创建更复杂的 Windows 界面。等价的控件也可以用在 Web 应用中，但是大多数基于 Web 的电子商务应用希望采用更简单的格式。我们准备详细讨论这些 GUI 控件，只是把它们总结如下：

- 下拉式日历：图 15-7a 演示了一个域，单击挨着日期的向下的箭头可出现图 15-7b 中的弹出式日历。我们熟悉的日历是模仿设计方式的另一个例子。
- 滑动编辑日历：这个输入控件提供了一种非数字方式用于选择值。
- 屏蔽编辑控件：这个输入控件实现了前面描述的格式检查。
- 省略控件：单击“三个点”图标激活一个用于数据录入的弹出式对话框，它可以用于输入由几个部分组合成的域（例如一个地址——街道、城市、州和邮政编码）。
- 另一种数字旋转器：这是另一种类型的输入旋转器。
- 因特网超链接：在功能上类似于按钮，超链接可以链接到 Web 页面，但也可以链接到其他 Windows 表单。这是一种隐藏相关输入表单的有效方式，这些输入表单并不用于所有用户或大部分用户。



a)



b)

图 15-7 高级 GUI 输入控件

- 组合复选框：这个控件用于在有几个方框可用的情况下组合几个复选框。
- 树形复选框：这个控件用于表现需要层次地组织成树状结构的数据选项。

15.3 如何设计和原型化输入

如何设计联机输入？过去，设计人员关心输入屏幕的整体内容、表现和功能——相对于需要设计的其他屏幕，设计人员只需设计一套菜单屏幕，然后用户通过从中选择一个菜单项进入相应的输入屏幕。很简单！但是，对于如今的图形环境来说，更强调开发一个整体的并可以很好地嵌入到用户工作环境中的系统，因此很少使用层次式菜单驱动的应用界面，这种界面以传统的老式文本或命令行应用系统为特征。

下面将演示如何完成输入设计的第一阶段，将利用音阶公司案例研究的例子进行说明。我们将介绍客户/服务器模式的基于 Windows 的输入，以及基于 Web 的运行在浏览器上的电子商务输入。后面（见第 16 章）我们将把输出（见第 14 章）和本章的输入集成为一个完整的用户界面和对话框。

15.3.1 输入设计和原型化的自动化工具

以前，输入设计的主要工具是记录布局图和显示布局图。如今，这种“草图”方式已经不常用了。如果不使用如今首选的原型化技术和快速应用开发策略（这些技术和策略使用自动化工具加速输入设计过程），输入设计将是一个很麻烦的过程。

在自动化工具出现之前，分析员只能绘制粗略的输入草图，以便感觉系统用户想让输入看上去怎样，或者批记录将如何组织。使用自动化工具，可以开发出更现实的原型。

最常用的输入设计自动化工具无疑是 PC 数据库应用开发环境。因为 Access 对于开发绝大多数企业级应用来说功能不够强大，所以你可能会惊奇如此多的设计人员使用 Access 设计这类应用的原型系统。给定一个数据结构（在 Access 中很容易定义），你可以快速地为输入数据生成或创建表单，可以使用本章描述的大部分 GUI 控件，然后用户可以试用这些表单，并告诉你使用情况（什么可行以及什么不可行）。

许多 CASE 工具提供了用于报告设计、屏幕设计以及原型化的工具，这些工具都使用需求分析期间创建的项目资料库。System Architect 的屏幕设计工具在前面的第 14 章（见图 14-6）中演示了。

大多数基于 GUI 的编程语言（例如 Visual Basic）可以方便地用来构造非实用的输入原型，这句话的关键词是非实用的（nonfunctional）：这些表单将看上去很真实，但是没有代码来实现任何按钮或数据域，这就是快速原型化的本质。

15.3.2 输入设计过程

输入设计过程并不复杂，其中有些步骤是基本的，而另一些则根据情况而定，具体步骤如下：

1. 确定系统输入并检查逻辑需求。
2. 选择合适的 GUI 控件。
3. 使用下列工具设计、验证和测试输入：
 - a. 布局工具（例如，手绘草图、打印机/显示布局图或者 CASE）。
 - b. 原型化工具（例如，电子表格、PC DBMS、4GL）。
4. 如果需要，设计源文档。

在下面的几个小节中，将介绍这些步骤，并演示一些来自音阶公司案例项目中的例子。

15.3.2.1 第 1 步：确定系统输入并检查逻辑需求

输入需求应该在需求分析期间就已经定义了。物理数据流图（或者设计单元，见第 12 章）是输入设计的良好起点，那些 DFD 确定了系统的从外部代理到过程的净输入和实现方法。

根据系统开发方法学和标准，每个净输入数据流也可以用数据字典或资料库（见第 8 章）中的一个逻辑数据流描述，其数据结构说明了要包含到输入中的属性或域。如果那些需求以关系代数的记号定义，你就能够快速确定哪些域重复出现，哪些域具有可选值等。考虑下面的数据结构：

数据结构定义逻辑需求	注 解
ORDER = <u>ORDER NUMBER</u>	← 输入的唯一标识符
+ ORDER DATE	← 必须取一个值的多个域之一，没有括号表示必须有值
+ CUSTOMER NUMBER	
+ CUSTOMER NAME	
+ CUSTOMER SHIP ADDRESS = ADDRESS >	← 到相关定义的指针
+ (CUSTOMER BILLING ADDRESS = ADDRESS >)	
+ 1 { PRODUCT NUMBER +	← 重复 1 ~ n 次的一组域，括号指示可选值
QUANTITY ORDERED } N	
+ (DEFAULT CREDIT CARD NUMBER)	← 一个可选的域，意味着一个不一定有值的域

如果没有这种精确的需求，则可能需要使用需求分析期间创建的需求原型。无论如何，你都应该以某种格式提供良好的需求陈述。

检查了音阶公司案例研究的需求分析期间定义的输入需求以后，可以确定有三个输入同主题 VIDEOTAPE 相关，设计决定使用一个输入屏幕支持这三个输入：NEW VIDEO TITLE、DISCONTINUED VIDEO TITLE 和 VIDEO TITLE UPDATE，系统应该收集这三个输入的数据内容或显示以下数据：

PRODUCT NUMBER +
UNIVERSAL PRODUCT CODE +
QUANTITY IN STOCK +
PRODUCT TYPE +
MANUFACTURER'S SUGGESTED RETAIL UNIT PRICE +
CLUB DEFAULT UNIT PRICE +
CURRENT SPECIAL UNIT PRICE +
CURRENT MONTH UNITS SOLD +
CURRENT YEAR UNITS SOLD +
TOTAL LIFETIME UNITS SOLD +
TITLE OF WORK +
CATALOG DESCRIPTION +
COPYRIGHT DATE +
CREDIT VALUE +
PRODUCER +
DIRECTOR +
VIDEO CATEGORY

属性 PRODUCT NUMBER、MONTHLY UNIT SALES、YEAR UNIT SALES 和 TOTAL UNIT SALES 不用由用户输入，由系统自动生成。而且，对于 TITLE COVER 属性，用户只需指定一个位图文件就可以了，这个位图文件包含了视频盘的实际图像。

15.3.2.2 第 2 步：选择合适的 GUI 控件

现在了解了要输入的内容，我们就可以设计适用于屏幕上显示的每个属性的屏幕控件了。首先将使用基于资料库的编程方法查看是否已经做出了这样的决策，是否存在其他属性特征，以及它们是否已经记录在资料库条目中。如果资料库中存在相应数据，只需复用对应输入屏幕上的属性的资料库条目；如果不存在相应的资料库条目，就需要创建它们。

为了给属性选择正确的 GUI 控件，首先必须检查每个属性的可能值。下面是对前面步骤中确定的输入属性的初步决策：

- PRODUCT NUMBER、CURRENT MONTH UNITS SOLD、CURRENT YEAR UNITS SOLD、TOTAL LIFETIME UNITS SOLD、UNIVERSAL PRODUCT CODE、MANUFACTURER'S SUGGESTED RETAIL UNIT PRICE、CLUB DEFAULT UNIT PRICE、CURRENT SPECIAL UNIT PRICE、

PRODUCER 和 DIRECTOR 属性都具有在范围上无限或者不可编辑的输入数据值。因为设计人员不能给用户提供一个有意义的选择值，所以选用单行文本框。由于属性 CATALOG DESCRIPTION 也满足这个准则，所以选用多行文本框（是指一些产品的备注框）。

- PRODUCT TYPE、LANGUAGE、VIDEO ENCODING、SCREEN ASPECT 和 VIDEO MEDIA TYPE 属性都具有有限的预定义值集合，所以单选按钮是这些输入项最合适的屏幕控件。
- CLOSED CAPTION 属性如何确定？它是一个包含 Yes/No 值的输入属性，所以选择复选框作为这个属性的 GUI 控件。
- 属性 QUANTITY IN STOCK、RUNNING TIME、COPYRIGHT DATE 和 CREDIT VALUE 包含了可以按照一种预期方式排序的数据值，因此，带有一个相关文本框的滚动框将是这些属性的合适选择。
- 属性 VIDEO CATEGORY 和 VIDEO SUBCATEGORY 具有大量的预定义值。由于有如此多的属性要在屏幕上显示，所以下拉式列表是最佳选择。
- 属性 TITLE COVER 是一个有趣的挑战。它的值实际上是一个视频盘封面位图文件的驱动器、目录和文件名，这个属性将使用一种称为图像框的高级控件存储视频盘封面图片。当用户选中这个对象时，系统将使用一组 GUI 控件和特殊对话框（用户交互）收集该条目的输入，我们将在后面第 3 步中演示这个输入。

需要重申的是，还有许多其他的屏幕控件可以用于输入数据，这个例子仅仅介绍了最经常使用的 GUI 控件。设计任务完成得如何将取决于你对这些常用 GUI 控件和其他高级 GUI 控件的熟悉程度。

15.3.2.3 第 3 步：设计、验证和测试输入

这一步为用户检查和测试开发原型屏幕。根据用户的反馈可能需要回到第 1 步和第 2 步，增加新的属性，并研究其特征。

让我们看一些音阶公司案例的屏幕原型。图 15-8 表示了用于处理 NEW VIDEO TITLE、DISCONTINUED VIDEO TITLE 和 VIDEO TITLE UPDATE 的一个原型屏幕。显示在屏幕右上角的标识是公司的一个标准所规定的——所有屏幕必须显示公司标识。由于我们决定将三个输入组合到一个屏幕上，所以在屏幕右上角和右边还增加了一些按钮，用来让用户选择期望类型的输入。我们将在第 16 章讨论这些按钮、命令、导航控件以及其用法。

注意图 15-8 中的以下注解：

- ① PRODUCT NUMBER、MONTHLY UNIT SALES、YEAR UNIT SALES 和 TOTAL UNIT SALES 被一种特殊的颜色屏蔽，这种方式作为一个视觉线索告诉用户这些域被锁住了，不能向其中输入数据，这些域的数据由系统自动生成。显示在屏幕上的其他域具有白色背景，这告诉用户它们可以被编辑。
- ② 为这些输入域指定了编辑掩码。UNIVERSAL PRODUCT CODE 域在指定的位置上包含了连字符，用户不用实际地输入这些连字符。相反，用户只需键入数字，之后整个内容会按照指定的编辑掩码重新显示。MANUFACTURER'S SUGGESTED RETAIL PRICE、CLUB DEFAULT UNIT PRICE 和 CURRENT SPECIAL UNIT PRICE 域也是这样。例如，用户在这三个域的任何一个中键入数字 9，按回车键，然后内容将会按照编辑掩码重新显示成具有美元符号和小数点的形式。

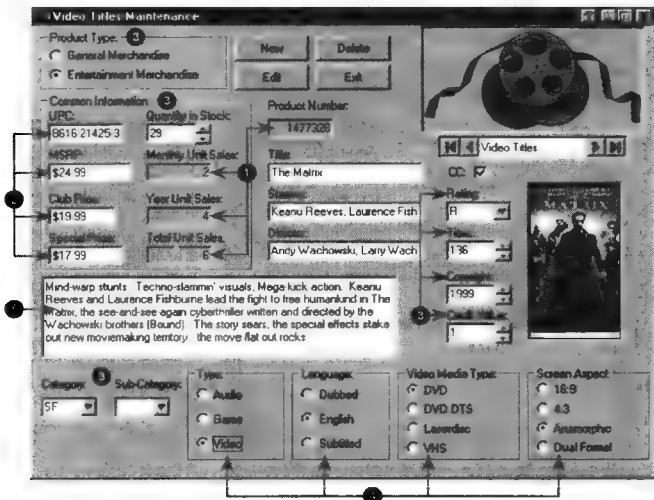


图 15-8 用于视频盘维护的输入原型

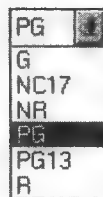
● 屏幕上的每个域都被赋予一个对用户有意义的标签。用户的反馈指出“CC”是“closed caption”的一个常用的简写。而且，用户指出 CATALOG DESCRIPTION 不需要标签。

① 相关的单选按钮被安排在一个带有描述性标签的组合框中。组合框经常用来关联一些相关的控件。例如，在标签为“Common Information”的组合框中的域被组合在一起，因为用户关联这些域到任何类型的 SoundStage 产品。另外，对应单选按钮的每个标签不是实际输入和存储在数据库中的数据，你所看到的是数据值的含义，而实际存储的值是一个编码。例如，如果用户为属性 LANGUAGE 选择标签为“English”的单选按钮，编码值 E 将实际地存储而不是“English”。

② 多行文本框有一个垂直滚动条特征，这说明在 CATALOG DESCRIPTION 域内还有其他文本没有显示。

当原型化输入屏幕时，让用户使用或测试屏幕很重要，用户测试应该包括演示用户如何获得合适的帮助或指示。新版本微软产品使用“工具提示”提供显示在一个屏幕上的按钮和方框的简要描述。当用户把鼠标放到某个对象上时，工具提示就会显示。另外，F1 键被广泛接受作为启动上下文敏感帮助的快速键。帮助按钮则是另一种提供系统帮助的方式。无论你使用哪种方式，都不需要在原型中实际地实现帮助。

最后，除非用户提出要求（或者由用户动作触发），否则原型不必对用户显示所有细节。例如，“Motion Picture Association of America RATING Code”的下拉式列表只显示一个默认值。但是，向下的箭头说明存在一个包含了可能值的列表框，这个列表框可以通过简单地单击向下的箭头看到，这个动作的结果演示见右图。



上一个例子相当简单，因为它仅仅包含了可以在一个数据库表中修改完成的数据。但是如果一个输入包括了要修改多个数据库表的数据怎么办（假设表之间有一个一对多关系）？请考虑 MEMBER ORDER 表，它与 MEMBER ORDERED PRODUCTS 表之间具有一个一对多关系，我们如何设计一个输入来为这两个表收集数据？

图 15-9 表示了在一个表单中输入 MEMBER 和 MEMBER ORDERED PRODUCTS 的原型屏幕。表单被分割成两个窗口区，MEMBER 数据在顶部窗口区，MEMBER ORDERED PRODUCT 数据在底部窗口区。你可能奇怪如果 MEMBER ORDERED PRODUCTS 的数量超过了分配给那个窗口区的空间怎么办，换句话说，底部窗口区的滚动条在哪里？许多 Windows GI 控件是“智能的”，如果底部窗口区中的行数超出了空间限制，就会自动地出现一个垂直滚动条。

作为最后一个 Windows 例子，图 15-10 表示了一个合并了数据流图中的 3 个不同或者类似输入的屏幕设计：NEW MEMBER、MEMBER CANCELLATION 和 MEMBER UPDATE，这个表单也使用了本章中讨论的标准输入控件。逻辑和物理数据流合并成单个屏幕设计的情况很常见。

图 15-9 会员订单的输入原型

图 15-10 会员购物的输入原型

15.3.2.4 第 4 步：如果需要，设计源文档

如果要用源文档收集数据，还必须设计文档，并提供给系统用户使用。在其最简单的形式中，源文档的原型可能是一个简单的草图，或者是一个效果图。

一个设计良好的源文档将被分成不同的区域。有些区域用于标识：包括公司名称、表单名称、办公表单编号、最后修改日期（一个经常被忽略的重要属性）和公司标志。其他区域包含确定表单的具体数据，例如表单顺序号（可能预先打印）和日期。文档中最主要的部分用来记录事务数据。出现一次的数据和重复的数据应该被逻辑地区分开。总数应该放在表单底部，因为它们通常是计算出来的，所以不用输入。许多表单都包含一个认证区域，用来供签名用。表单的指示应该放在容易看到的地方，最好不要在表单背面。

原型化工具近年来变得更加先进。电子表格程序（例如 Excel）可以做出十分逼真的表单模型。这些工具能使你很好地控制字体样式和大小以及公司标志图形。激光打印机可以生成原型的很不错的打印输出。

15.3.3 基于 Web 的输入和电子业务

我们想讨论的最后一个输入设计问题是基于 Web 的输入。音阶公司案例项目将各种电子商务和电子业务功能添加到会员服务信息系统中，这些功能中有些功能要求必须设计基于 Web 的输入。

项目的一个逻辑输入需求是基于 Web 的会员订单（MEMBER ORDER）。我们刚刚演示了客户/服务器版本的会员订单，现在看看基于 Web 的版本。可能经常需要实现一个 Web 商店前台（见图 15-11）。除了给会员提供关于音阶公司产品（一个输出）的信息以外，会员可以单击“BUY”按钮启动一次购买操作，这将会把会员带到购物推车屏幕（见图 15-12），购物推车已经成为电子商务应用中的一个公共用法。Web 界面往往比 Windows 界面更加艺术化，也许这就是它吸引人之处。在没有口头推销的情况



图 15-11 Web 购物推车的输入原型

下，界面需要可视地吸引客户购买产品，见图 15-12，请注意以下注释：

- ❶ 购物车帧独立于一般的导航帧（在左边），后者允许用户查询和浏览整个网站，并寻找其他要添加到购物车中的商品。
- ❷ 按钮、文本框、超链接、下拉框和其他常用控件在这里应用于 Web 界面，而不是 Windows 界面。
- ❸ “Checkout”超链接把会员送到下一“页”，以完成结账事务。

Web 界面具有一些优点，例如用户可以使用向前按钮和向后按钮浏览网站上不同的库存和订单页。

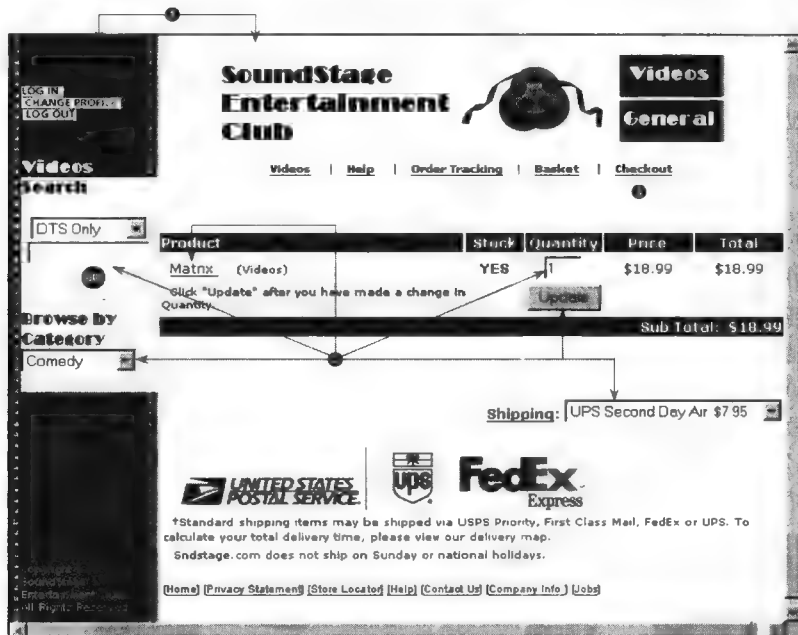


图 15-12 Web 购物车屏幕

复习题

1. 输入设计的目标是什么？
2. 源文档和数据条目之间的关系是什么？
3. 数据输入后的下一步工作是什么？这一步可以使用什么方法，它们在时效方面有什么不同？
4. 本书描述了哪些不同的输入方法？
5. OMR 和 OCR 之间有什么不同？
6. 对于生物识别输入，数据是如何输入和处理的？
7. 为什么智能卡技术能够存储大量的信息？举一些智能卡应用的例子。
8. 为什么人的因素在输入设计中很重要？在输入设计中需要考虑什么原理？
9. 有哪些技术用于验证数据？
10. 在什么情况下我们应该选择使用复选框，什么情况下使用单选框？
11. 下拉列表和组合框之间有什么相似之处？
12. 本书中建议了哪些先进的输入控制？
13. 输入设计过程有哪些步骤？
14. 什么是设计良好的源文档？
15. 当设计本书中建议的输入时，Web 界面相比 Windows 界面要面对哪些挑战？

问题和练习

1. 一个快餐连锁店的所有者雇用你的公司设计一个方法以更快地给顾客提供订单，使用较少的人力，但不降低质量。目前，快餐店使用传统的方法，让顾客排队等候下订单和付款；然后订单被打印并交给食物准备人员。你认为什么技术可以用于满足这些目标？
2. 看看以下的数据输入屏幕部分，公司的技术人员用来从公司的仓库订购部件。这个输入屏幕有什么设计错误吗？
输入技术员#: 输入技术员姓名:
输入部件#: 输入部件描述:
3. 销售终端（POS）——例如那些在 ATM、加油站

和商店中使用的——由于其方便性和多功能性变得十分常见。但按照人机界面设计的原则，它们的输入方法有时还有一些可改进的地方。你认为哪些方面需要改进？

4. 回答以下的判断题：根据需要解释你的回答。
 - a. 系统用户往往会被数据输入码搞混，经常输入错误的编码；因此，应该避免使用它们。
 - b. 批处理仍是可行的数据处理方式。
 - c. 在数据正确性方面，数据发起点和数据收集前的时间长度之间很少存在关联。
 - d. 为了优化使用 Windows 或 Apple 操作系统的个人计算机上的图形用户界面，发明了计算机鼠标。
 - e. 使用基于隐喻的屏幕设计被认为是太“做作”而且不专业。
 - f. 复选框最好只在少量的以前定义的没有共性的值时使用。
5. 考虑你使用过的、听说过的或者制作的最好的和最差的数据输入屏幕。使用你自己的经验，以及本章的内容，列出至少 5 个你认为重要的数据输入屏幕需求和/或原理（除了问题 3 中的那一个以外）。解释你为什么选择它们。
6. 你正在为一个客户治疗数据系统设计一个输入屏幕，它将在一个县使用。酒精和药物治疗提供者将输入关于它们的客户的治疗数据，然后把它发送到县里的行为健康部门。系统将在一个使用 DBMS 的客户/服务器网络上运行。业务规则要求所有数据项一个条目；没有可选域。数据将包括字母数据、用于计算的数字数据和日期等。有些数据域之间存在相互依赖关系，例如一个域是性别，另一个域是客户是否怀孕。唯一的客户标示号将用作键值，并由系统生成。

你应该在哪里设计数据输入控制和编辑，在网络的客户端还是服务器端？为什么？应该包括什么编辑和验证检查？

7. 匹配第 1 列中的定义或例子与第 2 列中的词汇：

A. 磁条卡	1. 滑动编辑日历
B. 带两个垂直对齐按钮的单一文本框	2. 数据捕捉
C. FedEx 包裹和发送数据处理方法	3. OMR
D. 同圆圈关联的值选择的文本描述	4. Quicken
E. 语音识别系统	5. Microsoft Visual Basic
F. 新数据的确定和获取	6. ADC 技术类型
G. 最容易造成数据输入错误的输入设备	7. 检查数字格式
H. 用于记录商业事务的纸质表单	8. 单选按钮
I. 基于隐喻的屏幕设计的例子	9. 远程批处理
J. 用于目标问题测试的光标记表单	10. 生物识别 ADC 技术
K. 用于值选择的非数学方法	11. 源文档
L. 决定主键数据输入错误的技术	12. 键盘
M. 基于资料库的编程方法的例子	13. 滚动框

8. 文本框可能是 GUI 界面中最常使用的数据输入控制。系统设计人员应该遵循什么规则和指南，当设计包括文本框的输入屏幕时？
9. 在决定为每个要捕捉和输入的数据属性使用最合适的 GUI 控制时，你应该自问什么基本问题？提供数据属性的例子，匹配数据属性和在何种场合下最合适的 GUI 控制的指令。
10. 基于常用的、容易辨识的隐喻的设计通常较易被系统用户接受，特别是初学者，因为它们的熟悉性提高了易用和用户友好的成分。你的公司将替换它的纸质电话消息表单成电子版本的，可以作为电子邮件的附件发送。为一个通常的纸质电话消息表单创建一个隐喻式的屏幕设计。（提示：你需要一个屏幕设计工具，但也可以在 Microsoft Word 或 Excel 中设计，创建你的模板，而不是使用一个常见的模板。）

项目和研究

1. 正如本书提到的，许多企业曾经雇用大量的职员输入数据。随着个人计算机和联机计算变得更加普遍，系统用户开始承担起数据输入的责任，而且在多数企业中数据输入职员的比例迅速地衰减。如今，因特网的爆炸性增长对大多数大型公司和政府部门的组织结构有着类似的巨大影响。
 - a. 在 20 世纪 80 年代至 90 年代，研究数据输入责任的转换——从数据录入职员到系统用户。讨论这些企业及其雇员所面对的问题。
 - b. 研究如今由于因特网促进了基于客户的数据输入的增长而发生的转变。这意味着什么？对公司的组织结构、雇员和客户有什么影响？
 - c. 研究你所在地区的大型公司或政府部门。比较它的 25 年前的组织结构、10 年前的结构和如今的组织。你发现了哪些相关的变化？
 - d. 研究在商业和 IT 杂志上的文章，关于技术对私人 and 公共企业的组织结构的影响。对于在数据输入方面将发生的下一次转变有什么预测？

你认为这些预测会实现吗？

- e. 总体来说，这些变化对于我们更好还是更糟呢？
2. 语音识别技术相当频繁地被用于通过电话输入命令或者响应自动问题数据。而且，有些技术专家预测有一天语音识别将代替键盘进行数据输入；当前就有一些使用语音识别技术进行数据输入的应用。研究在语音识别技术方面最近的发展，并回答以下问题：
 - a. 你找到了什么文章？它们对语音识别技术持什么观点？
 - b. 语音识别技术当前处于什么状态？
 - c. 你认为语音识别技术已经最终成熟到了很快可以作为键盘数据输入的一种可行的替代技术吗？还是仍需要进一步的成熟？还是走上了技术上的死胡同？请解释原因。
 - d. 如果你访问过包含语音识别软件进行数据或文本输入的应用（或者下载一个免费版本，而不破坏任何版权或使用限制），试着使用它。你将如何评价它们？
3. 一些新出现的技术被称为自动数据捕捉（ADC）技术。本质上说，它们把人排除在数据输入循环以外。这些技术之一是无线频率标识（RFID）技术，该技术很快就变得很常用。在网上和/或你学校图书馆研究 RFID 技术及其商业应用。
 - a. 解释 RFID 设备如何工作。
 - b. 当前 RFID 在私人 and 公共领域如何使用？
 - c. 它们的主要优点是什么？主要缺点是什么？
 - d. 关于 RFID 技术有哪些社会、经济和政治方面的问题？

小型案例

1. 输入设计不仅影响一个系统的易用性，而且影响其安全性。寻找如何通过输入到文本框的字符造成系统崩溃的例子。寻找如何解决这些安全漏洞。准备一份关于这个材料的短文，并在课堂上汇报。
2. 在第 14 章中的小型案例 2 中，你创建了一个联机表单。你恰当地使用了各种输入方法吗，例如：输入框、复选按钮、下拉框等？你将修改什

e. 今后 10 年里 RFID 将会用在哪些应用中？

4. 一般来说，在大多数企业中很容易发现一个设计得很差的源文档或数据输入屏幕。联系一个非营利的或者类似类型的企业，并志愿检查它们的表单和数据输入屏幕，并重新设计其中的一到两个。
 - a. 描述你为其做志愿工作的企业。
 - b. 你在源文档或输入屏幕中发现了任何设计问题吗？描述这些设计问题。
 - c. 描述你重新设计的源文档和/或输入屏幕。如果可能，提供一个例子。
 - d. 你做了什么改变，在进行修改时你遵循什么过程？提供一个例子。
 - e. 在重新设计表单和/或屏幕时，你遇到了什么挑战？
 - f. 企业满意你的设计吗？它使用了新的表单和屏幕吗？
5. 你的 Web 设计公司为一个连锁超市开发一个简短的基于 Web 的调查表，提供顾客喜欢超市什么和不喜欢什么的信息。让每个商店的其中一个雇员在商店游走，带着笔记本电脑，随即地选择顾客调查他们的喜好。向顾客提问 3~5 个问题。调查数据将通过基于 Web 的应用被直接输入到笔记本电脑中。
 - a. 在设计调查表单之前需要涉及哪些高层考虑？
 - b. 你将在调查表中包含什么问题？
 - c. 你还将包含其他什么数据？
 - d. 设计一个表单的原型。
 - e. 这比你想象的容易还是困难？描述你没有预见到但碰到的任何挑战。

么？为什么？

3. 修改你在小型案例 2 中的建议。提交你修改前后的屏幕截屏。确保包含了你的表单的 URL，以便老师可以检查你的工作。
4. 研究盲人用户的输入方法。写一篇短文简要描述这些输入方法，以及你如何将它们集成到信息系统中。

团队和个人练习

1. 圆桌讨论：我们如何使（一般意义上的）信息系统和计算机更容易使用？举例说明。
2. 个人练习：到目前为止，你已经进行了大量的小组工作。想象最近一次某人发脾气或者厌烦。如果他们通过电子邮件交流的话，在生气和厌烦时说出的话是否被放大了？你是否认为当写作时我

们会对言语更加小心，或者当其他人不能读到我们的肢体语言时？

3. 团队或个人练习：据说节食会刺激创造力。你认为这是真的吗？如果是真的，当他们越来越物质上富足时，一个公司或者个人如何保持创造力和精神上的饥饿？

用户界面设计

本章概述和学习目标

在本章中，你将学到如何为一个系统设计用户界面。用户界面应该为用户提供友好的使用方式，通过用户界面用户可以同应用程序打交道，处理输入并获得输出。在第 14 章和第 15 章中，学习了如何设计输出和输入。用户界面设计和原型化关系到应用程序的整体表现，并可能要求重新修订那些初始的输出和输入原型。如今有两种经常遇到的界面：用于连接大型主机的终端（或者表现为终端的微机）和更常见的连接到微机的显示器。设计用户界面也存在几种不同的策略风格。本章介绍以下内容：

- 区分不同类型的计算机用户，并为每类用户进行设计。
- 辨识几个重要的人类工程学因素和指南，并将其纳入到用户界面的设计中。
- 集成输出设计和输入设计成为一个完整的用户界面，用户界面建立了用户与计算机之间的对话。
- 理解操作系统、Web 浏览器和其他用于用户界面设计的技术。
- 在信息系统中使用合适的用户界面策略。使用状态转换图来规划和协调信息系统的用户界面。
- 描述如何使用原型化技术设计用户界面。

本章关键术语

专家用户 (expert user) 是有经验的计算机用户。

初学者用户 (novice user) 是不太有经验的计算机用户。

对话 (dialogue) 是屏幕和消息的整个流程。

分页显示 (paging) 一次显示一满屏字符。

滚动显示 (scrolling) 在屏幕中向上或向下移动显示信息，一次移动一行。

功能键 (function key) 是用于编程特殊操作的一组按键。

鼠标 (mouse) 是一种引起指针在屏幕上移动的设备。

菜单驱动 (menu driven) 是一种对话策略，要求用户从选项菜单中选择动作。

代理 (agent) 是可复用的软件对象，可以在不同的应用软件和网络之间运行。

状态转换图 (state transition diagram, STD) 是一种用于描述用户会话期间可能出现的屏幕的顺序和变化的工具。

16.1 用户界面设计概念和指南

在前两章中，我们讨论了输出设计和输入设计。在本章中，将把输出设计和输入设计集成为一个完整的用户界面，从而建立用户与计算机之间的对话。这种对话决定了系统的每件事情：从启动系统或者登录系统，到设置属性或偏好，到获得帮助。况且输出和输入本身也是界面的一部分。我们需要检查屏幕到屏幕之间可能的转换。在客户/服务器应用（例如基于网络的 Windows）和 Web 应用（例如基于因特网或内联网的浏览器）中，用户有许多种不同的方式使用菜单、超链接、对话框等，这使用户界面十分易用和友好，但同时也使设计和编程极大地复杂化了。

如今大多数用户界面通过快速地构造原型进行设计，这些原型使用快速应用开发环境生成，例如微软公司的 Visual Studio、Borland 公司的 JBuilder（用于 Java 语言）或者 IBM 公司的 Visual Age（用于各种语言）。这些原型一般功能不完整，但是它们确实提供了足够的功能来演示界面操作。当我们进入系统生命周期的构造阶段时，程序员和分析员将完成这些功能。

下面将首先介绍各种影响用户界面设计的用户、人的因素和人类工程学指南。

16.1.1 计算机用户的类型

“系统用户”大致可以分为专家和初学者，或者不可任意支配的和可任意支配的两类。

专家用户是有经验的计算机用户，他们花大量时间使用专门的应用程序，计算机的使用对他们来说通常被认为不是随意的。在大型主机计算时代，他们被称为专用用户。专家用户一般熟悉（但不一定精通）应用的操作环境（例如 Windows 或者 Web 浏览器）。他们已经投入了时间学习使用计算机，并会继续投入时间克服不太友好的用户界面。一般地，他们会记住日常的操作，以便不用再查找指令，或者不需要使用详细的计算机反馈和指示，他们希望以尽可能少的动作和按键完成任务。

初学者用户（有时称为偶然的用户）是不太有经验的计算机用户，他们一般较少使用计算机，或者甚至是偶然地使用，计算机的使用对他们来说可以看成是随意的（尽管这种情况越来越少）。简单地说，初学者用户需要得到比专家用户更多的帮助。帮助有很多种形式，例如菜单、对话框、提示和帮助屏幕。大多数管理人员（尽管他们的计算机知识不断增长）都属于初学者之列，他们是来识别和解决问题、探索机会以及制定计划和管理期望的——而不是来学习和使用计算机的。计算机只是现代管理人员的工具，当需求出现时，他们希望尽可能快地实现其效益，然后继续工作。

专家用户和初学者用户实际上是所有用户中的两个极端。完全没有使用过计算机的初学者正变得越来越少，所有专业的大学课程都要求学习计算机知识，而且所有专业的学生都已经发现了增加跨学科计算机专业知识（有时称为信息学）的价值。初学者用户通常也可通过实践和经验成为专家用户。因特网的净社会影响就是更多的人越来越了解计算机，每过一年就产生出一些“半瓶子醋”的专家用户。用户界面设计正在向 Web 浏览器发展有什么奇怪呢？甚至在 Windows 应用中也是这样。

很难想象如今的年轻学生和专业人士不熟悉计算机。无论如何，如今绝大多数系统仍是为初学者用户设计的，但适应了专家用户的要求，其中的焦点是用户友好性或人类工程学。

16.1.2 人的因素

在设计用户界面之前，理解那些经常使人们觉得计算机系统难用的因素很有好处。我们喜爱的用户界面设计专家 Wilbert Galitz（见“推荐读物”）提出了以下几个界面设计问题：

- 过多地使用计算机行话和缩写。
- 不明显或者不太直观的设计。
- 不能确定执行的动作（我下一步做什么）。
- 不一致的问题解决方式。
- 设计不一致。

按照 Galitz 的说法，这些问题导致了混淆、惊慌、挫折、厌倦、误用、放弃和其他不期望的后果。

为了解决这些问题，Galitz 提出了下列最重要的用户界面设计原则：

- 理解你的用户及其任务。当我们扩展信息系统，使其可以通过因特网实现企业到客户（B2C）和企业到企业（B2B）功能时，理解用户及其任务就变得越来越困难。
- 让用户参与界面设计。找出用户在他们现在的应用中喜欢什么和不喜欢什么，并让他们从一开始就参与屏幕设计。使用如今的 PC 数据库和快速应用开发技术很容易实现这条原则。
- 在实际用户中测试系统。观察和聆听是这里的关键技能。通过初始培训之后，尽量避免过多地培训和强制用户学习系统的使用。相反，观察他们的操作和出现的错误，并听取其评论和提出的问题，这样才能更好地理解他们同用户界面的交互。
- 进行迭代设计。第一个用户界面可能不太令人满意，任何用户界面设计都可能需要通过多次设计迭代和测试。界面什么时候才算设计完成呢？可能永远不会完成！但 Galitz 建议一个合适的目标是：95% 的典型用户（不管他们是初学者还是专家）能够进行有目的的工作（不管是日常的还是不太常用的），而没有困难或者不需要帮助。

16.1.3 人类工程学指南

确定了用户类型之后，设计中还应该考虑一些重要的人类工程学因素：

- 系统用户应该总是知道下一步干什么。系统应该总是提供指示，说明如何继续前进、后退、退出系统等。有几种情况需要某些类型的反馈：
 - 告诉用户系统现在期望什么。这可以采取一种简单消息的形式，例如“准备好”、“请输入命令”、“选择一项或多项内容”或者“请输入数据”。
 - 告诉用户数据已经正确地输入了。这可以通过简单地将光标移动到表单的下一个输入域，或者是显示一条消息，例如“数据输入正确”。
 - 告诉用户数据没有正确地输入。最好使用简单的消息解释正确格式，帮助功能可以用更详细的指示和例子来补充说明这些消息。
 - 向用户解释延迟处理的原因。有些动作需要几秒或者几分钟才能完成，例如排序、索引、打印和修改。请使用简单的消息告诉用户系统并没有失败，例如，“正在排序中——请等待”，或者“正在检索中——可能需要几分钟”，或者“请等待”。Windows 的“沙漏”或者 Internet Explorer 的“转动地球”都是表示处理正在进行的图标线索。
 - 告诉用户某个任务完成了或者没有完成。这在延迟处理的情况下特别重要，但在其他情况下也重要。显示一条消息就够了，例如“打印完成”或者“打印机没有就绪——请再试一次或者联系网络管理员”。
- 屏幕应该被格式化，以便各种类型的信息、指示和消息总是出现在通常的显示区域。这样，系统用户就大约知道在哪里可以找到特定信息。在大多数窗口环境中，设计标准经常会规定状态消息或者弹出式对话框窗口的显示位置。
- 消息、指示或者信息的显示时间应该足够长，以便系统用户有时间阅读。大多数专家建议重要的消息应一直显示到用户确认了它们之后。
- 少使用显示属性。如果过多地使用显示属性（例如闪烁、加亮和反转显示），就会分散注意力，相反，明智地使用则有助于提起对某些重要事物的注意力——例如，下一个要输入的域、一条消息或一个指示。
- 应该指出用户要输入的默认值和默认答案。在窗口环境中，有效值经常作为一个滚动区域在独立的窗口或者对话框中显示。默认值（如果可用）通常应该位于第一个，并且被明显地突出显示。
- 预测用户可能犯的错误。甚至在给出了最明确的指示时，系统用户仍可能会犯错误。如果用户可能会执行一个危险操作，就要让他知道他在干什么（例如，可以显示一条消息或者一个对话框：“你确实要删除这个文件吗？”）预防总是比不预防好！
- 如果出现错误，不应该允许用户不改正错误就继续操作。应该显示关于如何改正错误的提示（或者例子），错误可以使用声音或者颜色提醒，然后在一个弹出式窗口或者对话框中解释，也可以定义一个“帮助”选项触发显示其他提示信息。
- 如果用户做了某些可能是灾难性的事情，应该锁住键盘以防止进一步的输入，并且应该显示一条提示信息让用户通知分析员或者技术支持人员。

16.1.4 对话语气和词汇

屏幕和消息的整个流程称为对话。对话的语气和词汇是用户界面设计中十分重要的因素。关于对话的语气，我们提供以下指南：

- 使用简单且语法正确的句子。最好使用口语，而不要使用正式的书面语。
- 不要逗笑或者装腔作势。当某人每天要用系统 50 次时，幽默很快就会变得毫无意义。
- 不要故作谦卑。不要低估系统用户的智力，例如：不要提供反复的赞扬或者表彰。

至于计算机对话中使用的词汇，下面的建议可能会有用：

- 不要使用计算机行话。
- 尽量避免使用缩写词。缩写词的使用是以假设了用户理解如何翻译它们为前提，所以应该首先检查一下是否真的如此。

- 使用简单的词。使用 NOT CORRECT 代替 INCORRECT，这样可以减少误读或误解的机会。
- 词汇的使用应该保持一致。例如，不要使用 EDIT 和 MODIFY 来指代同一动作。
- 注意短语的用法——使用合适的行为动词。下面的建议应该有所帮助：
 - 当指一系列选项时，使用 SELECT 或者 CHOOSE，不要用 PICK。注意指出用户可以从可选列表中选择一个值还是多个值。
 - 为了要求用户输入特定的数据或指示，使用 TYPE 而不用 ENTER。ENTER 这个词可能会与回车键混淆。
 - 使用 PRESS，而不用 HIT 或者 DEPRESS，指出需要键盘动作。只要可能，用实际打印在按键上的符号或者标识符。例如，“↵”键在一些键盘上指示 RETURN 或 ENTER 键。
 - 当指示一个屏幕上的鼠标光标时，使用 POSITION THE CURSOR，而不要用 POINT THE CURSOR。

16.2 用户界面技术

如今绝大多数用户界面是图形方式的，图形用户界面（或者 GUI）的基本结构要么在计算机操作系统中提供，要么在所选的因特网浏览器中提供。在客户/服务器信息系统中，客户端用户界面在 PC 操作系统上执行；在因特网和内联网信息系统中，用户界面在 PC 的 Web 浏览器上执行（浏览器在 PC 操作系统上执行）。

16.2.1 操作系统和 Web 浏览器

如今客户端计算机（在客户/服务器网络中）上的主流 GUI 操作系统是微软公司各种版本的 Windows，苹果公司的 Macintosh 和各种形式的 UNIX（包括 Linux）也占有一些市场份额。对于不断增长的手持式和膝上型客户端计算机，当前的主流操作系统是 Palm 公司的 Palm OS，微软的 Windows Mobile 也占有一定的市场份额。

操作系统越来越不是用户界面设计中的关键技术因素。因特网和内联网应用在 Web 浏览器中运行，而大多数浏览器可以在多种操作系统中运行，这使得设计出一个较少依赖计算机本身的用户界面成为了可能。这种计算机平台无关性的优点应该是显然的：你不再需要为每种计算机平台和操作系统编写用户界面，而只需为一个或两个浏览器编写用户界面。在出版本书时，主流的 Web 浏览器是 Internet Explorer 和 Firefox，但是浏览器内部的版本问题仍然存在。

除了操作系统和浏览器，用户界面的整体设计还可以通过用户的显示器、键盘和指点设备的可用特征得到增强或者受到限制，下面就简要地介绍其他方面的一些设计因素。

16.2.2 显示器

显示区域的大小对用户界面设计很重要。不是所有的显示器都是 PC 监视器。一些非 PC 终端依然存在。终端是仅仅显示由远程计算机（通常是一个大型主机）传送的数据和信息的非 PC 显示器。虽然许多终端已经被 PC 代替了，但是用户仍然经常被迫使用终端模拟器同遗留的大型主机应用程序打交道。终端模拟器在屏幕上打开一个窗口，仍然按照以前的 Windows 终端方式显示信息和指令。对于这些终端和终端模拟器，两种最常见的显示区域是 25 行 80 列和 25 行 132 列。

幸运的是，个人计算机监视器已经代替了绝大部分终端，而且大部分较新的应用系统和再工程应用系统都正在采用图形界面。对于 PC 监视器，我们不按照行和列来度量显示区域。虽然对角线度量（例如英寸）经常被引用，但更好的度量是图形分辨率。图形分辨率用像素度量，也就是显示在屏幕上的离散光点数量。如今一个 17 英寸的对角显示器最常见的分辨率是 1 024 000 水平像素 × 800 000 垂直像素，更大的显示尺寸支持更多的像素；但是，设计人员设计用户界面时一般应该使用最低的常见或合理的分辨率。

显然，掌上电脑和笔记本电脑以及专业化终端显示器（例如收银机和 ATM 中的显示器）只支持小得多的显示器，这些因素都必须在用户界面设计中考虑。

显示区域显示给用户的方式由显示器的技术能力和操作系统的能力共同控制。分页和滚动是两种

最常用的显示方式。**分页显示**一次显示一满屏字符，整个显示区域称为一页（或一屏），页按需由下一页或前一页替换——很像翻一本书的书页。代替分页显示的另一种常用方式是**滚动显示**。滚动显示在屏幕上向上或向下移动被显示的信息，一次移动一行。这类似于电影的最后字幕滚过屏幕的方式。需要重申的是，PC 显示器提供了更广泛的分页显示和滚动显示功能。

16.2.3 键盘和指点设备

绝大多数（但不是所有的）终端和监视器都集成了键盘，明显的例外是笔记本电脑，例如 PalmPilot。键盘的主要特征包括字符集和功能键。

大多数 PC 键盘的字符集十分标准，这些字符集可以用软件来扩展以支持额外的字符和符号。对于专用的终端或工作站，厂家可能设计定制键盘。大多数键盘包含称为功能键的特殊键，PC 键盘通常有 12 个这类功能键，终端已知包括了 32 个功能键。**功能键**（通常标记为 F1、F2 等）可以用来在用户界面中编程某种公共的重复操作（例如，HELP、EXIT 和 UPDATE）。在操作系统中，功能键经常被预定义，但是应用开发人员可以为特殊的系统定制它们。功能键的使用应该保持一致，也就是说，任何信息系统的程序应该一致地使用相同功能键实现同一功能。例如，在操作系统和应用程序中，F1 通常用作帮助键。

大多数 GUI（包括操作系统和浏览器）使用指点设备，例如鼠标、笔和触摸屏。显然，最常用的指点设备是鼠标。鼠标是一个小的手掌大小的设备，放在靠近终端的平面上，其下面有一个小的滚动球。当在平面上移动鼠标时，就引起指针在屏幕上移动。鼠标上的按钮可以用来选择光标移动到其上的对象或者命令。由于需要滚动 Web 页面和文档，许多鼠标现在都包括了一个轮子，这使得用户可以更方便地滚动页面和文档，而不需要再使用滚动条进行滚动了。

笔在使用手持式设备（例如 PalmPilots）的应用系统中越来越重要。因为这类设备经常不包含键盘，所以用户界面可能需要设计成允许在屏幕显示的键盘上“按键”，或者使用手写技术，例如 Graffiti 或者 Jot，一些预先构建的组件可以实现这些公共特征。

如前所述，最常见的用户界面是图形用户界面——要么基于 Windows，要么基于 Web 浏览器。本章剩下的部分将专门介绍图形用户界面设计。

16.3 图形用户界面风格

用户界面设计就是说明系统用户和计算机之间的对话（或者交谈），这种对话通常导致数据输入和信息输出。图形用户界面存在几种不同风格，这些风格过去被看作是互相替代的，但是现在它们正逐渐融合。本节简要介绍几种用于设计图形用户界面的风格或策略，以及它们如何应用到如今的应用程序中，我们将用常用的软件应用程序演示这些风格。

16.3.1 窗口和框

基于操作系统的和基于浏览器的 GUI 的基本结构是窗口。窗口是一个有边界的矩形区域，标题（以及备选的文件名）显示在每个窗口的顶部。

窗口可以比监视器实际显示的可视区域小或者大，通常在右上角具有标准的 GUI 控件，用来最大化自身到显示屏幕的大小，最小化自身到一个图标（在屏幕底部），恢复到以前的大小，以及退出系统（或者关闭窗口）。

显示在一个窗口中的文件、表单或者文档可以适应（或者可以不适应）窗口的大小。当文件、表单或者文档超过窗口大小时，就使用窗口右边和底部的滚动条浏览，并指出光标相对于整个文件、表单或者文档的当前位置。

窗口可以被分成称为框的区域。每个框可以独立于窗口中的其他帧动作，并独立地使用特征（例如分页、滚动、显示属性和颜色），也可以被定义用于不同用途。框在 Windows 和 Web 浏览器中都很常见。

在一个窗口或框中，可以使用前面两章中使用的所有用户界面控件（例如文本框、单选按钮、复选框、下拉式列表、按钮等）。另外，本章后面将介绍许多其他类型的用户界面控件。

最后，窗口经常在底部显示一个任务条或者托盘，这个任务条可以用于显示消息、进度或者特殊工具（在后面讨论）。

16.3.2 菜单驱动界面

最老最常用的对话策略是菜单选择。初学者用户和专家用户使用不同类型的菜单。菜单驱动策略要求用户从选项菜单中选择动作。

菜单驱动的对话实际上比 GUI 出现得要早。图 16-1 演示了一个典型的 GUI 以前的层次式菜单，它逻辑地将菜单选项组合成高级选项以简化表现。如图 16-1 所示，如果主菜单选项 DISPLAY WARRANTY REPORTS 被选择，子菜单 WARRANTY SYSTEM REPORT MENU 就将出现；然后，如果 PART WARRANTY SUMMARY 选项被选择，报告定制和报告屏幕就顺序地显示。对于菜单可以嵌套到多深的层次没有技术限制，但是嵌套越深，对于专家用户来说，就越需要有到深层菜单的直接通路，

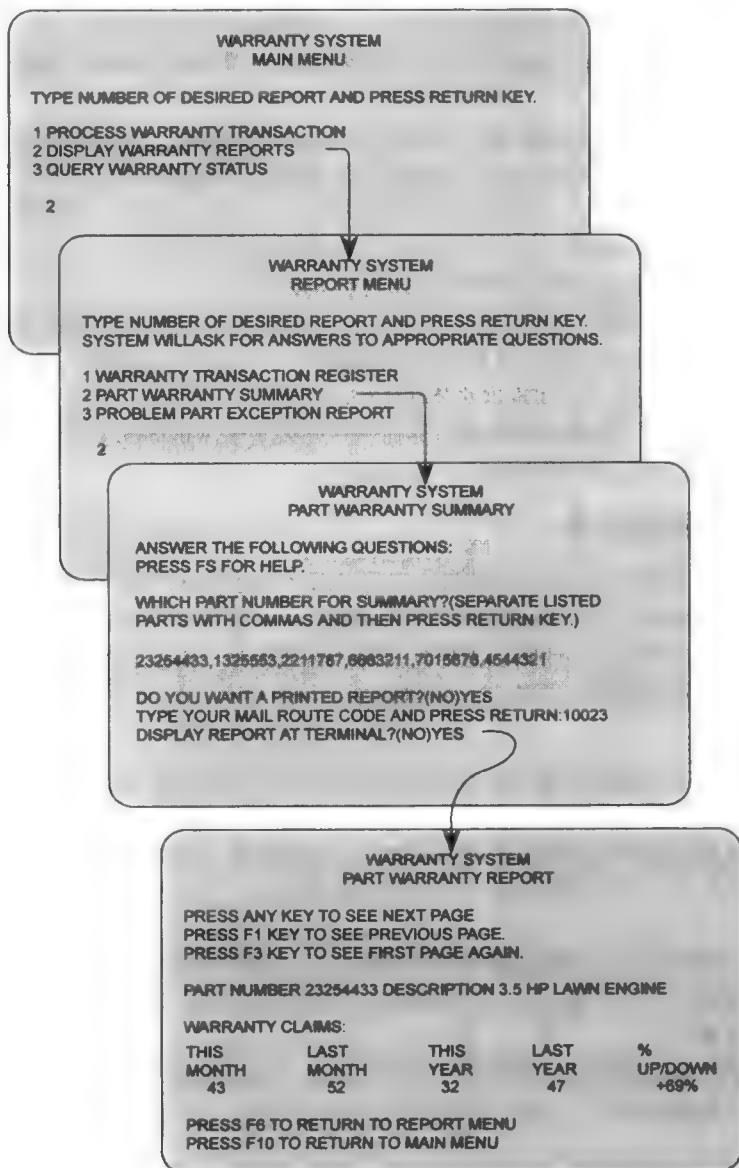


图 16-1 一个典型的层次式菜单对话

他们可能会觉得多级导航很讨厌（称为屏幕失效）。而且大多数用户也需要有直接回到主菜单或者高级子菜单的通路，而不是沿着原路返回。

GUI 以前的层次式菜单相对容易设计，例如，图 16-2（来自本书的早期版本）显示的对话图用来映射屏幕到屏幕的转换，并确保一致性和完整性。但图形用户界面的出现使得菜单设计变得极其复杂。

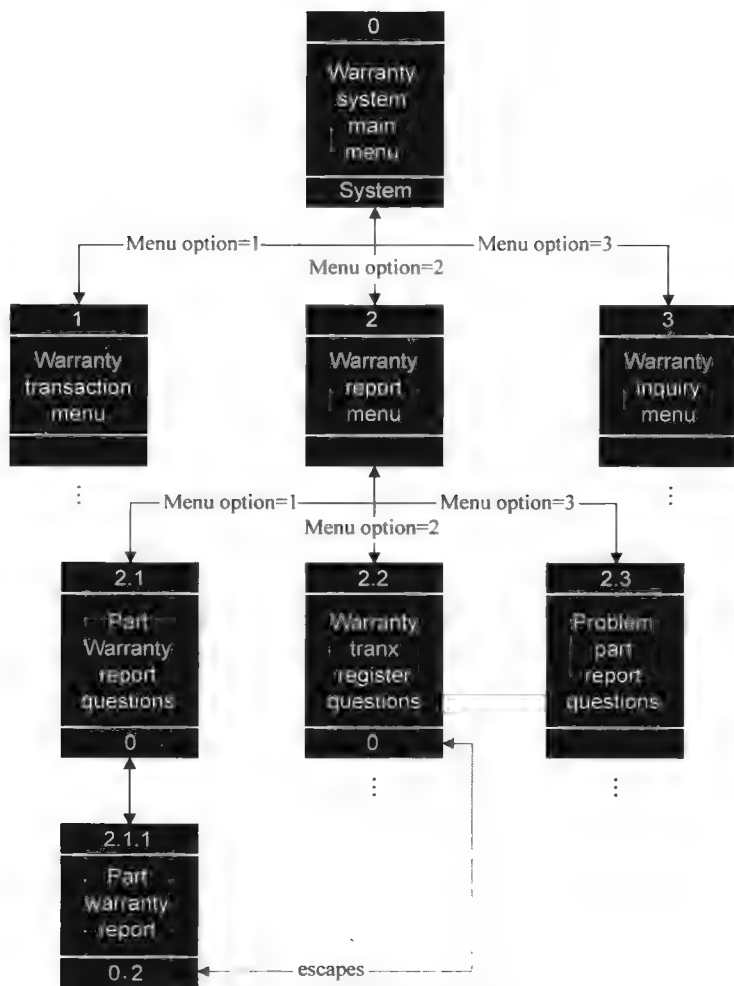


图 16-2 简单的对话图

16.3.2.1 下拉式菜单和层叠式菜单

在一个 GUI 中，菜单通常来自菜单条的下拉式菜单和层叠式菜单实现。如图 16-3a 所示，每个菜单选项实际上是一组相关的命令和动作。如右侧图显示了一个菜单模板。这些菜单选项中有许多常见于各种或者所有的应用中。例如，基于 Windows 的应用一般包括如下图所示的菜单组。

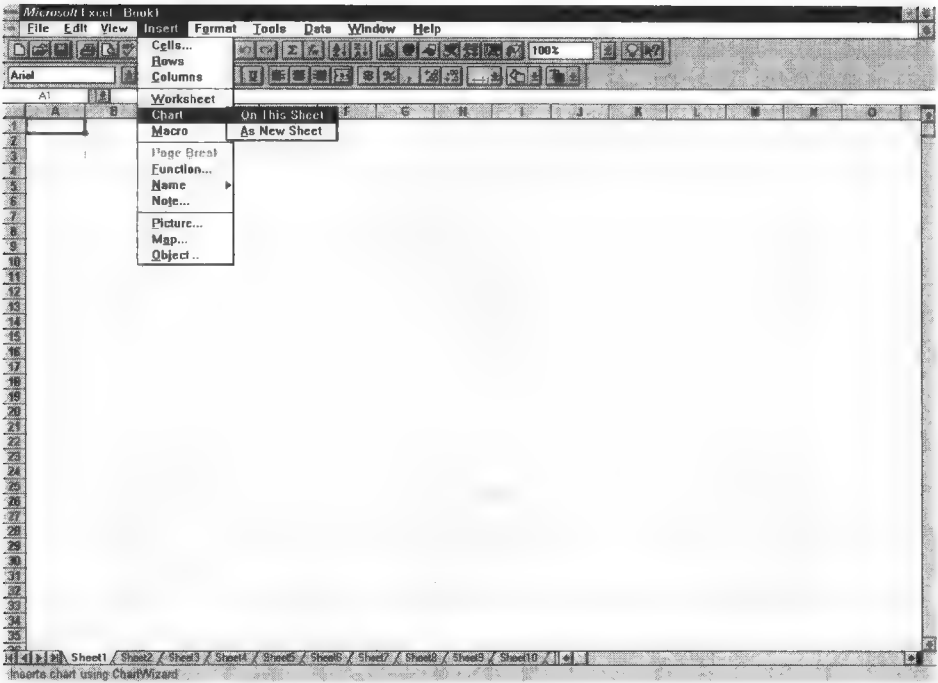
File Edit View Format Tools Actions Window Help

基于 Windows 的应用所包括的菜单组

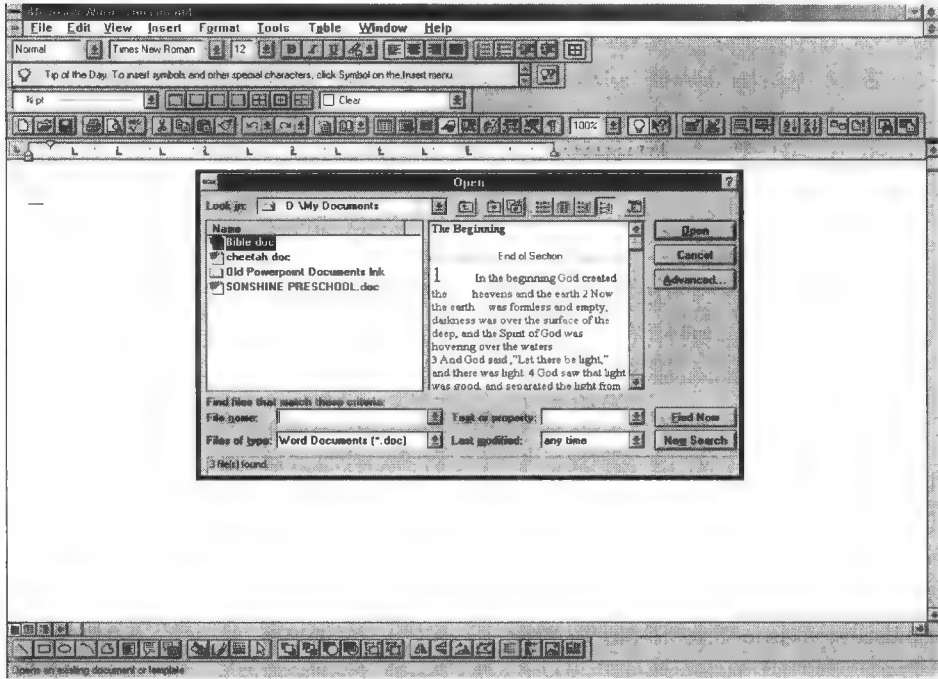
用户可以使用鼠标选择菜单组，或者使用键盘快捷键（例如，同时按下 Alt 键和带下划线的字母键，称为记忆键、快捷键或热键）选择菜单组。



菜单模板



a)



b)

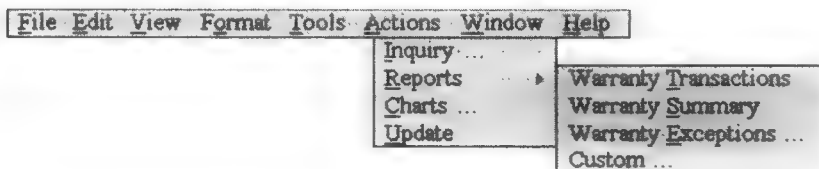
图 16-3 下拉式菜单和层叠式菜单以及对话框示意图
a) 下拉式菜单和层叠式菜单 b) 对话框

每个菜单组有它自己的下拉菜单。当用户从菜单条中选择某个菜单组时，一个子菜单就被拉下来，如下图所示。



下拉菜单

注意子菜单可以由水平线分成子组（例如，组合所有的“保存”或者“打印”子菜单命令）。在某些情况下，一个命名的子菜单动作后跟一个省略号（3 个点），表示将显示（弹出）一个对话框（窗口）（见图 16-3b），以表现额外选项或者收集额外指示。另一种情况是，一个命名的子菜单动作将用一个小箭头指示还存在另一个子菜单，这称为层叠式菜单，如下图所示。



层叠式菜单

16.3.2.2 浮动式菜单和弹出式菜单

不是所有的菜单都可以归入菜单条。有些 GUI（例如 IBM 公司的 OS/2 Presentation Manager 和 UNIX 的 OSF/Motif）允许浮动式菜单。使用浮动式菜单技术，用户可以选中某个下拉菜单或者层叠菜单，把它从菜单条拖开，然后重新定位到屏幕上的其他地方。如果必须一直使用菜单，这种方式特别有用。注意仅仅是原始菜单的一个拷贝被实际地撕了下来。

弹出式菜单是上下文敏感的，并且依赖于指点设备。它通过用户单击鼠标右键激活，菜单凭空弹出（见图 16-4），弹出哪个菜单取决于光标在屏幕上的位置。光标可以指到一个空白区域、一个域、一个单元格、一个字或者一个对象，右键单击将显现一个菜单，而菜单中仅仅包括那些应用于光标当前位置的动作——所以才被称为上下文敏感。弹出式菜单也可以层叠。因为对弹出式菜单的出现没有提供视觉线索，所以它们主要供专家用户使用。

16.3.2.3 工具条和图标菜单

工具条由图标（图片）构成，图标代表了动作和命令的快捷方式，它们通常嵌入在下拉式菜单和层叠式菜单中（见图 16-5）。在 Windows 应用中，一个包含常用动作的工具条位于菜单条的正下方，用户可以不使用菜单而直接单击这些工具或图标立即调用相应动作。任何应用程序中都可以创建工具条，应用开发人员可以为用户提供定制工具条的一定的灵活度。

虽然大多数工具条位于菜单条的正下方，但是许多应用程序允许工具条被定位到窗口的左边、右边或者底部，或者任何用户觉得方便的地方，这个动作称为停放工具条。而且，一些工具条可以做成在窗口内任何方便的地方浮动（或者移动）。

注意 在基于 Web 的应用中，工具条由浏览器提供，而且不能为专门的应用进行定制。浏览器上最重要的图标是 PAGE FORWARD、PAGE BACKWARD 和 HOME PAGE 图标，它们对所有的因特网和内联网浏览都是标准的。

图标菜单在窗口的主体部分使用图片表示菜单选项。在 Windows 应用中，这些图标菜单经常用来为计算机应用提供一个主功能和活动的控制中心，或者记录使用计算机应用的业务步骤。图 16-6 演示了图标菜单，其中每个按钮都代表一个直观的菜单选择。

图标菜单在基于 Web 的应用中很常见，因为那些应用系统运行在浏览器中——浏览器不允许开发人员修改浏览器菜单条上的菜单命令。Web 应用经常使用可单击的图片、图标和按钮代表菜单选项。

Web 类界面的流行对 Windows 用户界面设计影响很大。大多数客户/服务器信息系统的客户端用户界面都是模仿用户最常用的 PC 工具，例如字处理和电子表格。用户熟悉那些工具，所以设计其他的应用仿照那些菜单、工具条、对话框是有意义的。但是基于 Web 应用的流行产生了新的消费者风格界面。

像 Web 页面一样，Windows 应用的消费者风格界面更加艺术化。虽然仍然使用菜单条，但是窗口看起来更像一个 Web 页面，因此也就显得更友好。界面包括可单击的图标和按钮，它们代替了更传统的 Windows 菜单方式。只要图标和按钮不过于复杂，对 Windows 应用来说，这样会具有比传统应用（例如微软公司的 Office 和 SmartSuite）中看到的更友好的外观。图 16-7 演示了一个消费者风格的 Windows 界面。

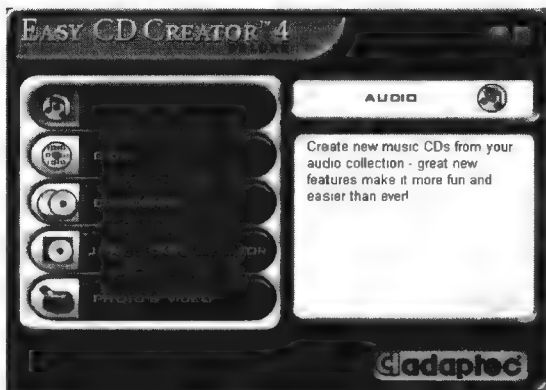


图 16-6 图标菜单

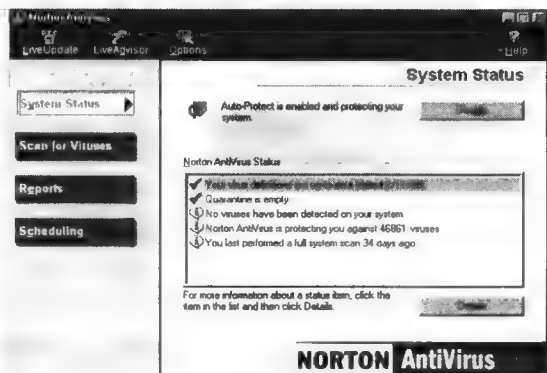


图 16-7 消费者风格界面

16.3.2.4 超文本和超链接菜单

超文本和超链接是现代 Web 用户界面的产物。最初创建超文本和超链接是用来在 Web 页面和网站内部以及网站之间进行导航。词、词语或者短语被标记为超链接（通常格式为带下划线的文本，而且通常具有不同颜色），单击超链接将把用户导航到相关页面（或者页面内的书签）。

可以方便地扩展和调整这项技术以实现基于 Web 的因特网和内联网应用系统的菜单。因为这些应用在浏览器中运行，而且因为浏览器的菜单条和命令是固定的，所以不能像我们在 Windows 应用中所做的那样方便地定制菜单。相反，可以使用超文本和超链接在 Web 页面的内容中实现那些菜单。每个菜单选项都是一个超文本短语（或者一个超链接图标或按钮），它们调用了其他 Web 页面上的动作或表单。从本质上说，这种方法创造了类似于图 16-1 中介绍的层次式菜单结构。这有点讽刺意味，因为 Web 应用系统的菜单设计正使用一种传统的设计风格，而这种风格在遗留的大型主机应用中得到广泛采用！

超文本和超链接不再是因特网和内联网应用的专利，许多现代的 Windows 应用已经通过实现混合 Windows/Web 用户界面采用了 Web 技术。例如，图 16-8 演示了 Intuit 公司的 Quicken 系统（流行的个人财务程序）的用户界面。由于它有许多超链接，所以这个系统看起来就像一个 Web 页面，虽然它确实包括了许多可选的 Web 特征，但它实际上是一个 Windows 应用。它也有自己的 Windows 菜单条，并同所有 Windows 应用公用的定制下拉式菜单和层叠式菜单保持兼容。随着企业采用因特网和内联网作为所有信息系统的基础，我们预期这种混合界面将会越来越普及。

16.3.3 指令驱动的界面

除了菜单以外，有些应用系统还使用基于指令集（也称为命令语言界面）的对话。因为用户必须

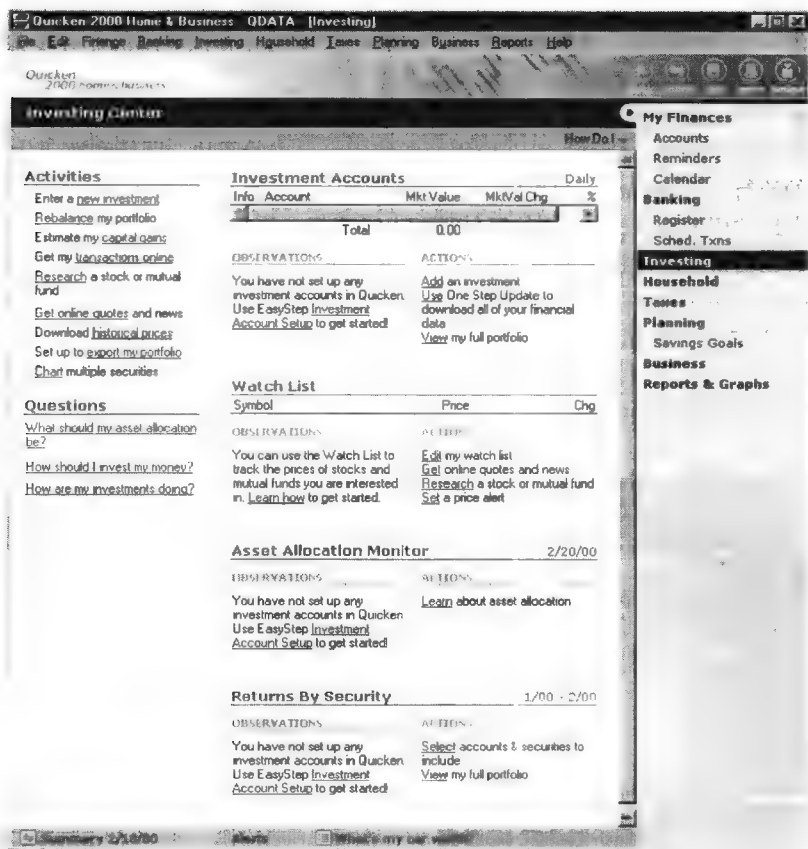


图 16-8 混合 Windows/Web 界面

学习指令集的语法，所以这种方法最适合于专家用户使用。有以下三类语法可定义，应该使用哪类语法取决于可用的技术：

- 基于语言的语法围绕一个广泛接受的命令语言构造，用户可以使用它调用相应操作。例如 Query by Example (QBE) 和结构化查询语言 (SQL)，二者都是数据库语言，都可以用于访问数据并创建定制报告。
- 助记语法围绕为定制信息系统应用定义的命令构造。系统提供给用户一个屏幕控制台，在其中用户可以键入调用动作的命令，并响应计算机输出。理想情况下，命令应该对用户有意义（包括任何允许的缩写）。
- 自然语言语法使用户可以用他们的自然语言输入问题和命令。系统按照已知的语法解释这些命令，如果不能理解用户想要什么，系统就请求用户澄清。

指令驱动的界面风格常见于遗留的大型主机应用和早期基于 DOS 的 PC 应用，但是这种交互风格仍然可以在如今的图形应用中发现。例如，Access 数据库包含了一个查询工具，它允许开发人员可视化地（指点）开发数据库查询（见图 16-9）。开发人员只需选择一个查询中包含的数据库表、列和行，如图 16-9a 所示，然后，如果愿意的话，可以查看和编辑实现这个查询的命令级 SQL 代码，如图 16-9b 所示。需要重申的是，指令集方法要求用户拥有一定程度的专业知识和经验。

16.3.4 提问-回答对话

提问-回答对话风格主要作为菜单驱动对话或者指令驱动对话的补充——系统提问问题，用户回答问题。最简单的问题涉及 Yes 或者 No 回答，例如：

你想看到所有部件吗？[NO]

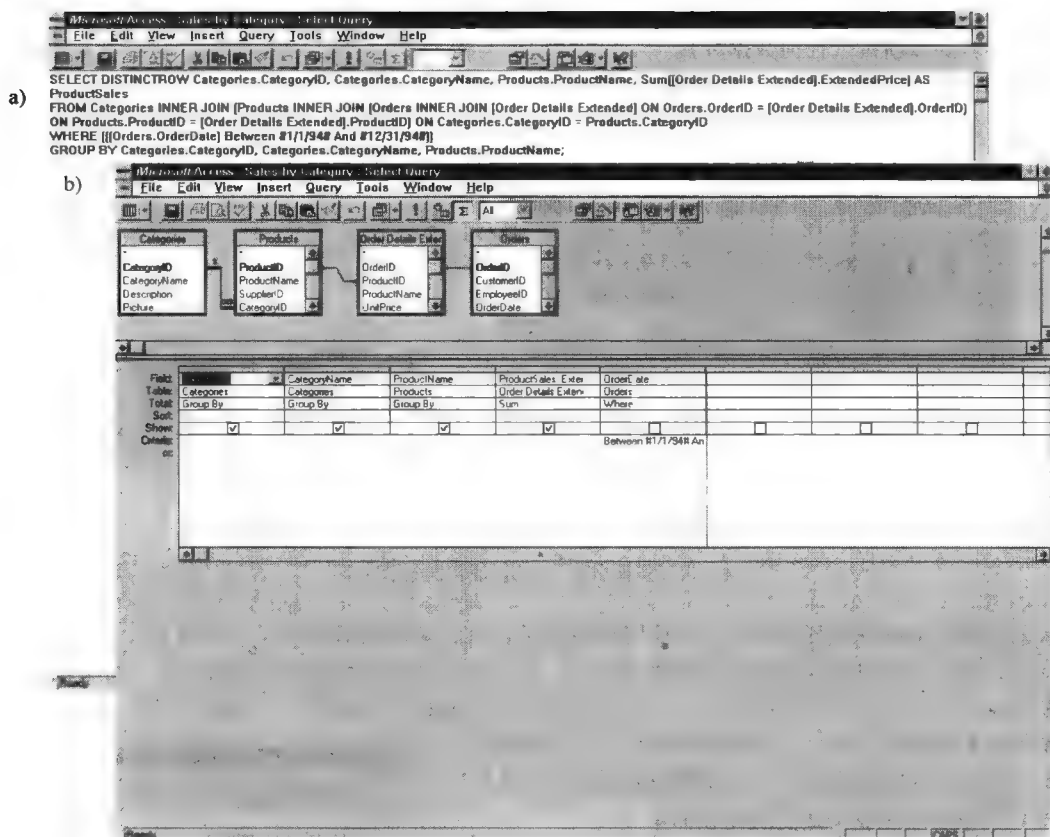


图 16-9 指令驱动界面

注意如何给用户提供默认答案。问题可以更加明确，例如，系统可能会问：

你对哪个部件编号感兴趣？

这种风格要求你考虑所有可能的正确答案，并处理如果用户输入了不正确的答案系统应该采取的动作。提问-回答对话很难设计，因为你必须试图考虑系统用户可能做错的每件事情。

提问-回答对话在 Web 应用中很流行。例如，一个轿车预约系统可能会问一系列问题来定义你预约哪种类型的轿车以及租金合同。

你想在哪里取你租的车？

你计划在哪里还你租的车？

取车的日期和时间？

还车的日期和时间？

你需要哪种类型和大小的车？

你有优惠券吗？

每个问题都可以包含一个下拉式答案列表供用户选择，这些问题和答案共同定义了业务事务。

16.3.5 用户界面设计的特殊考虑

除了确定用户界面风格以外，还需要考虑一些用户界面设计的特殊因素。用户将如何被识别和认证才能使用系统？用户界面中有哪些安全和保密考虑？最后，用户如何通过用户界面获得帮助？

16.3.5.1 内部控制——认证和授权

在大多数系统中，在系统用户被允许执行某些动作之前，他们必须被系统认证和授权。换句话说，系统用户必须“登录”到系统中。大多数登录要求同时使用“用户 ID”和“口令”，在理想情况下，

应该要求用户使用与其局域网账号相同的登录方式（Windows XP、NT 和 Windows 2000 允许使用这种认证，从而不需要重新键入任何数据域）。

图 16-10a 演示了音阶公司案例的用户登录界面，USER ID 和 PASSWORD 将根据网络账号文件进行认证。注意当用户输入口令时，口令显示成星号，这是一种常用的安全和保密保护措施。如果用户 ID 和口令没有通过认证，就会显示如图 16-10b 所示的安全授权失败对话框。

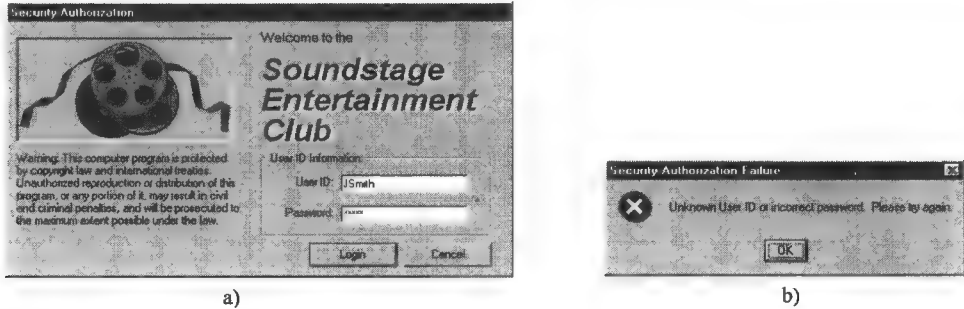


图 16-10 认证登录屏幕和错误显示屏幕

a) 认证登录屏幕 b) 认证错误显示屏幕

认证只是这个解决方案的一半。一旦用户通过了认证，系统就必须确定用户对这个系统的访问和服务特权。有许多模型可以用来确定和管理特权。一个重要的指南是分配特权到角色，而不是到个人。在大多数企业中，人们经常变换工作——他们被重新分配和提升到新的工作岗位和角色；而且，工作描述和角色不时变化；最后，人们离开企业，而且有些人会被解雇。由于所有这些原因，特权应该分配到角色，最后确定 USER ID 是否可以获得角色就比较简单。

对于每个角色来说，需要分配专门的特权。特权可以包括读专门的表或视图的许可，在专门的表或视图中创建、修改或删除记录的许可，生成和查看专门的报告的许可，执行专门的事务的许可，等等。尽管技术上这些内容不是界面的一部分，但是定义这些角色和许可可对设计一个合理的登录界面是必要的，同时从可使用性说明完整的系统认证和授权安全模型也是必要的。

随着电子商务的出现，客户和其他企业必须对相互声称的身份表示信任。客户可以通过因特网传输他们的信用卡号和其他私人信息。因此，音阶公司购买了一个 Web 证书向其俱乐部会员和潜在会员证实自己的身份。在任何时候，音阶公司会员都可以使用浏览器界面查看图 16-11 中的认证证书。通过这个证书，音阶公司 Web 网站将显示一个“安全服务认证”图标（见右图——挂锁），它告诉客户他们的数据将被加密，以确保其信用卡和个人数据传过网络时不会被他人拦截或者访问。

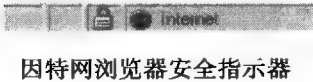
16.3.5.2 联机帮助

人们想要的是能够立即直接访问的上下文敏感信息，也就是说，帮助信息需要足够灵巧，能够想象出用户可能试图要做的事情。将帮助系统和培训教材直接构建在应用程序中是一个明显的趋势，因此联机帮助成了用户界面的一部分。

一个应用系统的通用帮助信息构建在 Windows 应用的“Help”菜单中。对于 Web 应用来说，帮助信息通常构建成为独立页面（通常在独立的窗口“弹出”页面），所以用户也可以仍然停留在启动帮助请



图 16-11 服务器安全证书



因特网浏览器安全指示器

求的页面上。

如今，HTML（超文本标记语言）正逐渐地成为构造图形用户界面——Web 应用和 Windows 应用帮助系统的通用语言。例如，微软公司的 Office 的整个帮助系统就是用 HTML 编写的。

如今的自动化工具简化了帮助系统的设计、构造和测试。完整的帮助系统包括一个目录、大量的提示、例子和一个全面的索引。许多帮助著作软件包（例如 Macromedia 公司的 RoboHelp）给作者提供了字处理工具以辅助一个完整帮助系统的计划、提纲、编写、索引和超文本链接。

设计良好的帮助系统将实现众多的帮助元素。也许最经常遇到的帮助类型是那些用户必须激活的帮助。如前所述，F1 功能键几乎被普遍接受为帮助请求命令。在大多数基于 Windows 应用系统（商业的或者定制的）中，系统经常使用标准的帮助菜单条组织和表现不同类型和级别的帮助。最后，如图 16-12 所示，Windows 界面和 Web 界面都经常使用工具提示控件提供与特定的工具和对象图标关联的弹出式帮助。当用户有意地将光标放置在图标（或者对象）上时，就会显示工具提示。所有的图标都要使用工具提示，因为用户界面设计人员绝不能假设系统用户完全理解显示在图标上的图像或者标签的含义。

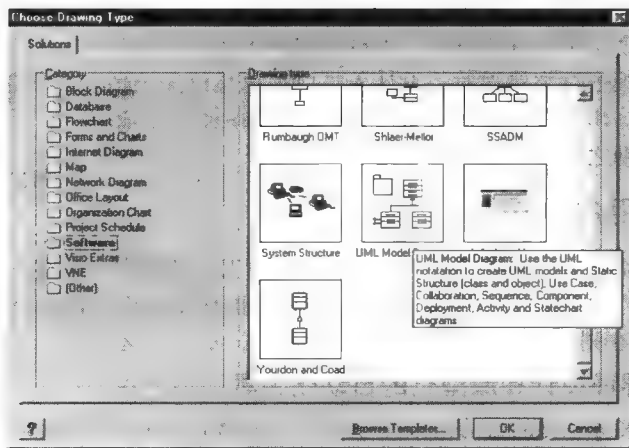


图 16-12 帮助工具提示

对初学者用户特别有用的另外两个也是常用的帮助特征是帮助向导和帮助代理（或者助手）。如图 16-13a 所示，帮助向导指导用户执行复杂的过程，显示一个要求用户输入和系统反馈的对话框序列。请注意以下内容：

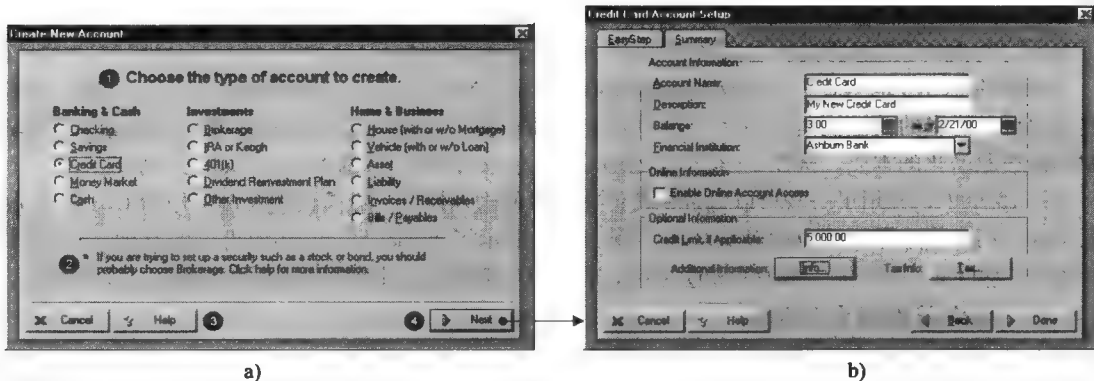


图 16-13 帮助向导

- ❶ 作为一个典型的帮助向导，对话通常包括一系列的指示或者问题供用户响应。
- ❷ 向导包含了解释，以辅助用户理解和做出决定。

① 向导也提供获取完成这个任务的更详细帮助的按钮。

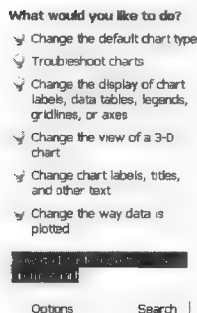
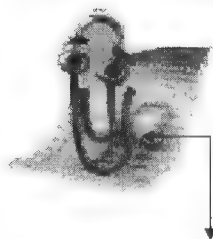
① “Next”按钮表明还有其他或者后续的步骤（一旦完成了一系列的对话框，“Next”按钮通常就转变成“Finish”按钮）。图 16-13b 表示了帮助向导支持的结果屏幕和后续步骤。

微软和第三方控件厂商销售向导以帮助开发人员构造向导。

代理是另一种应用于帮助系统的技术。代理是可复用的软件对象，可以用于不同的软件应用中，甚至可以通过网络运行。微软的帮助代理（称为助手）为所有 Office 应用程序提供了一种公共的帮助工具。其默认形式是一个模拟的纸夹（见右图）。微软公司的帮助代理可以编程到定制应用中，既可以用于 Windows 中，也可以用于 Internet Explorer Web 浏览器中。一个用户单击一下帮助代理就启动了帮助。

微软公司的帮助代理通过自然语言处理技术得到加强，用户可以用自然语言编写查询，然后由代理解释以显示最可能的帮助响应。然后用户可以从响应中选择一个，或者进入更详细的帮助索引。

设计一个优秀的帮助系统最重要是设计人员应该预测系统用户的错误。当设计用户界面报告这类错误时，设计人员应该总是给系统用户提供解决错误的帮助。离开任何帮助会话后，用户应该总是可以返回他们请求或者收到帮助之前的位置。



帮助代理/自然语言处理

16.4 如何设计用户界面

如今的图形环境强调开发一个能够很好地融合到用户工作环境中的完整系统。下面将演示如何为一个图形环境设计用户界面，仍然利用音阶公司案例研究的例子，将检查客户/服务器模式的 Windows 输入和基于 Web 的运行在浏览器中的电子商务输入。

16.4.1 用于用户界面设计和原型化的自动化工具

支持用户界面设计和原型化的自动化工具与第 14 章和第 15 章中介绍的那些输出和输入设计工具一样。最常用的用户界面设计自动化工具是 PC 数据库应用开发环境。大多数 PC 数据库产品（例如 Access）功能不够强大，不能开发绝大多数企业级应用，但是它们应付一个应用的用户界面屏幕的原型化是绰绰有余的。给定一个数据库结构（在 Access 中很容易定义），你可以快速地生成或者创建输入数据的表单，可以在界面中包括本章中描述的大部分 GUI 控件，然后用户可以使用那些表单并告诉你使用情况。

许多 CASE 工具还包括用于屏幕布局和原型化的工具，这些工具使用需求分析期间创建的项目资料库。System Architect 的屏幕设计工具在图 14-7 中已经演示过。

大多数基于 GUI 的编程语言（例如微软公司的 Visual Studio）可以方便地用于构造用户界面屏幕的非实用原型，这里的关键词是非实用。表单看上去很真实，但没有代码来实现任何按钮或者数据域，这就是快速原型化的本质。例如，图 16-14 演示了构造一个简单菜单的 Visual Studio 对话框。

在第 15 章中，介绍了一些可以包含在任何窗口中的输入控件。界面设计人员可用的 GUI 控件数量仅仅受限于用来构造界面的应用开发环境。图 16-15 演示了 Visual Studio 环境提供的其他一些 GUI 控件，包括：Outlook 工具条、具有标题的可排序的列表、步进器、目录式列表框和非输入式下拉列表。

16.4.2 用户界面设计过程

用户界面设计过程并不复杂，基本的设计步骤包括：

1. 以图表形式描述用户界面对话。
2. 原型化对话和用户界面。
3. 获得用户反馈。
4. 如果需要，回到第 1 步或第 2 步。

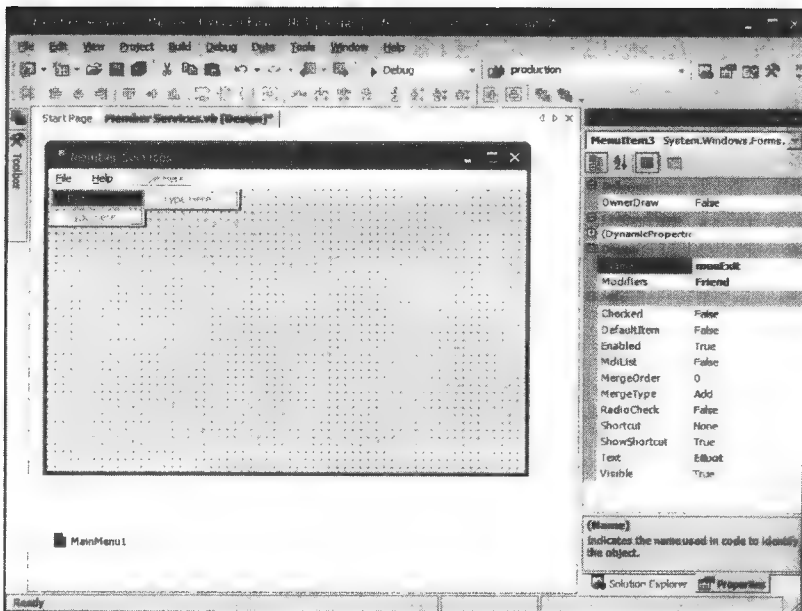


图 16-14 Visual Studio 菜单构造

实际上，上面的步骤在实践中并非严格按顺序进行。相反，各个步骤是迭代的。例如，随着原型的开发，系统用户检查原型，用户提供反馈，反馈可能要求修改或者生成一个新原型。下面将介绍一个迭代过程中的这些步骤，并演示来自音阶公司项目的一些例子。

16.4.2.1 第 1 步：以图表形式描述对话

一个典型的用户界面可能包含许多可能的屏幕（可能包含几个窗口），也许几百个屏幕。每个屏幕都可以被独立设计和原型化，但是这些屏幕之间的协作呢？

屏幕一般按照特定的顺序出现，但也可以在屏幕之间切换，而且某些屏幕可能只在某种情况下出现。

更困难的是，有些屏幕可能重复出现直到某些条件被满足为止。这听起来几乎像一个编程问题，不是吗？我们需要一个工具来协调可能出现在用户界面上的屏幕。**状态转换图**用来描述当系统用户坐在终端前时可能出现的屏幕顺序（作者正在按一般的意义使用屏幕这个词，当设计图形界面时，这个词可以指整个显示屏幕、一个窗口或者一个对话框）。可以把状态转换图看作是一个地图，每个屏幕对应一个城市，并非所有的道路都通向所有的城市，矩形用来表示显示屏幕，箭头表示控制流和引起屏幕成为活动的（或者接收焦点的）触发事件。矩形仅仅描述了对话期间可以显示什么，箭头的方向则指示了屏幕出现的顺序。因为不同的动作触发了来自一个给定屏幕的控制流或者到一个给定屏幕的控制流，所以每个方向都要绘制一个独立的箭头，每个箭头都有自己的标签。

下面看一下正在为音阶公司项目构造的一个对话（见图 16-16），该部分完成的音阶公司状态转换图是用一个 CASE 工具（Popkin 公司的 System Architect）开发的。请注意以下注释：

- ① 状态转换图包括对第 15 章开发的一些音阶公司输入屏幕的引用。
- ② 这幅图还包括对第 14 章开发的一些输出屏幕的引用。

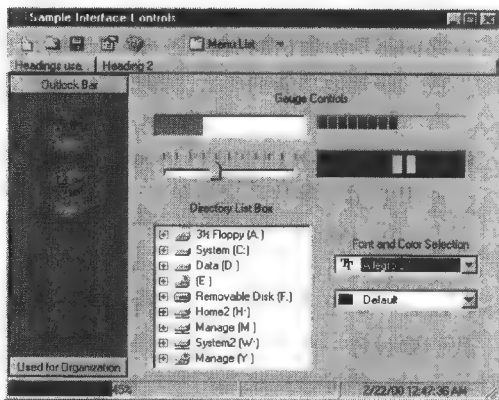


图 16-15 其他用户界面控件

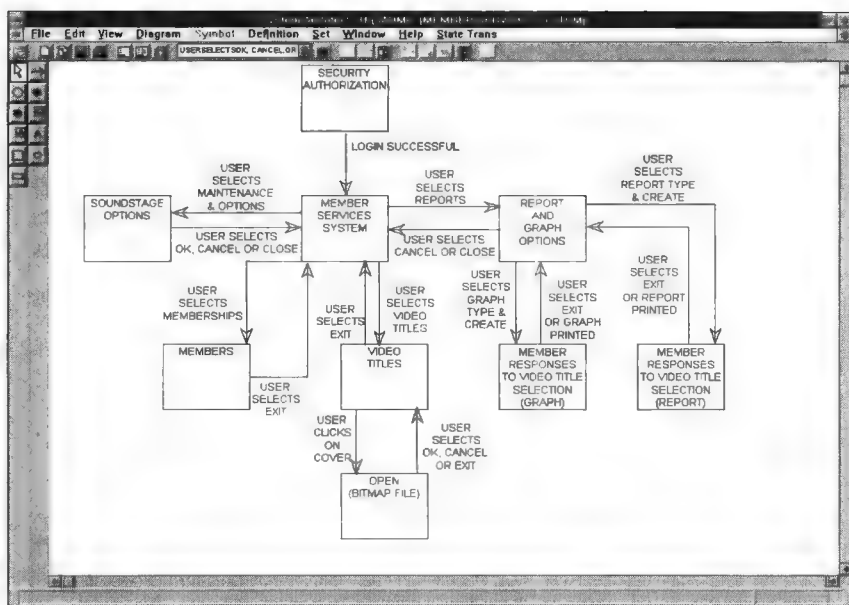


图 16-16 音阶公司项目的部分状态转换图

● **MEMBER SERVICES SYSTEM** 屏幕将是一个需要设计和原型化的新屏幕，它将作为应用的主窗口，在向用户提供访问前面设计的系统输入和输出屏幕的能力方面扮演重要角色，还将向用户提供完成一些其他功能的能力（超出输入和输出处理以外的，但通常在用户界面设计期间设计）。用户只有首先访问了 **SECURITY AUTHORIZATION** 屏幕并且成功地登录进系统时，才能访问系统。

● **SOUNDSTAGE OPTIONS** 屏幕是另一个需要创建的新屏幕，它使得用户可以设置各种会话期间要使用的用户参数和默认值，例如：选择一个打印机、缩放和其他参数。

图 16-16 的状态转换图可能会变得很大，特别是当所有的输入、输出、帮助和其他屏幕都添加到图中时。所以，设计人员经常把大的状态图分割成一套独立的更简单且更容易阅读的小图。

16.4.2.2 第 2 步：原型化对话和用户界面

我们要设计一些新屏幕。这些新屏幕有些用来把应用合成在一起，并且其输入和输出屏幕在前面设计了；另一些屏幕用来让用户定制应用系统的交互，使应用系统满足用户自己的喜好；还有一些屏幕用来处理系统控制，例如备份和恢复。

下面来看一些要为音阶公司会员服务系统创建的新屏幕。系统用户将首先看到本章前面讨论的认证登录屏幕（见图 16-10），按照状态转换图，用户登录成功后进入图 16-17 中描述的音阶公司会员服务系统主菜单。请注意以下说明：

- ① 用户及其访问特权得到证实。根据他们的访问特权，某些功能将被启用或者禁止。
- ② 通过一个菜单条或者通过一个垂直的菜单按钮，用户可以完成常用的会员服务业务操作。这些按钮将让用户使用前面设计和原型化的输入屏幕处理相应事务。按钮上使用了文本标签，因为分析员不能建立所有用户都能辨识的表示操作的图标（图片）。菜单条和按钮提供了热键，这使得用户可以通过键盘或鼠标进行选择。一个组合框表示了相关操作的按钮。
- ③ 用户具有完成各种日常维护操作的能力。

通过 **MEMBER SERVICES SYSTEM** 屏幕的菜单条，用户可以为工作会话设置参数，图 16-18 显示了这个新屏幕。请注意以下注释：

- ① 这个屏幕利用选项卡作为允许用户修改四套相关参数的方式。
- ② 用户使用滑动条控件调整后台查询的优先级。该控件经常用于以下这种情况：数据域的值可以表现为一个空间间隙形式，并且数据域只要有近似值（而不是精确值）就足够了。

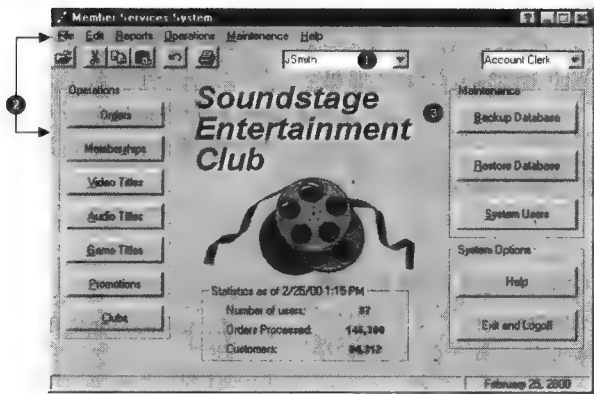


图 16-17 音阶公司主菜单

实际上,分析员将需要设计“General”、“Print”、“View”、“Database”选项卡的内容和表现。按照状态转换图,这个屏幕将返回控制到父窗口(MEMBER SERVICES SYSTEM)。

按照状态转换图,系统界面也为系统用户提供了说明报告定制选项的机会。图 16-19 描述了一个原型屏幕,音阶公司用户可以使用它选择一个特殊的报告(或图表),并定制报告内容。

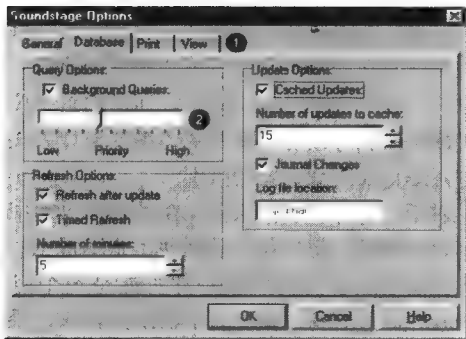


图 16-18 音阶公司参数和优先屏幕

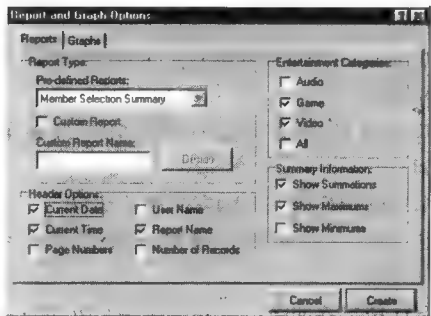


图 16-19 音阶公司报告定制对话框

请研究状态转换图和刚介绍的屏幕,了解整个系统对话的这一部分是如何工作的。通过研究屏幕的整个集合,可以发现需要对屏幕做一些修改,例如修改颜色、公共按钮和菜单选项的命名使其保持一致,以及就其他感官冲突问题进行处理。请遵循 GUI 应用应该遵守的任何标准。

16.4.2.3 第 3 步: 获得用户反馈

试验或者测试用户界面是本章中介绍的原型化环境的一个主要优点。试验或者测试用户界面表示在工作系统的广泛编程和实际实施之前,系统用户试验和测试界面设计,分析员可以观察这个测试以改进设计。

在缺少原型化工具的情况下,分析员至少应该与系统用户一起走查屏幕草图模拟对话过程。在用户界面设计中,用户反馈是很重要的,分析员应该鼓励用户参与测试应用程序的界面。最后,如果知道需要做出修改,分析员应该重新执行第 1 步和第 2 步设计步骤。

复习题

1. 系统用户为什么应该参与设计用户界面的过程?
2. 谁是专家用户? 他们为什么被称为专家?
3. 为什么有些用户界面会使用户觉得混淆或者困惑?
4. 实际的用户测试系统意味着什么?
5. 我们应该做什么以确保系统用户知道在系统中做什么?
6. 界面应该如何处理错误?

- 7. 按照计算机对话中使用的词汇，应该考虑哪些因素？
- 8. 为什么当设计应用时 Web 浏览器变得越来越重要？
- 9. 解释分页和滚动。
- 10. 当为应用设计功能键时，我们应该考虑什么？

问题和练习

- 1. 尽管菜单驱动的界面比 GUI 老，但仍很常见。菜单驱动的界面与 GUI 界面之间有什么主要差别？菜单驱动的界面有什么主要的优点？什么主要的缺点？
- 2. 回答下列判断题。根据需要解释你的回答。
 - a. 在屏幕对话中应该使用不同的动作动词来描述所需的键盘动作，以便增加变化和兴趣。
 - b. 大多数经理是专家用户，因为他们为了有效管理需要高级的 PC 专业知识。
 - c. 企业应该期望在最终结果使企业满意之前，专业设计人员（他们受到高度欢迎，而且报酬很高）多次细化和修改他们的用户界面设计。
 - d. Windows 用户界面设计经常借用 Web 界面风格和技术。
 - e. 应用软件需要不止一种类型的帮助菜单或对话。
 - f. 用户欣赏智能或者幽默的屏幕消息。
 - g. 设计用户界面的过程是直接的而且容易理解的。
- 3. 在设计用户界面时，必须考虑信息安全性和隐私性。描述当构造用户界面设计中的内部控制时必须考虑的一些设计指南和因素。
- 4. 匹配第 1 列中的定义或例子和第 2 列中的词汇。

A. 经常需要多级菜单驱动	1. 平台无关性
B. 应用软件的整个屏幕和消息的顺序	2. 消费者风格的界面
C. 用全屏方法来显示用户一次看到的区域	3. 终端模拟
D. 信息一次上移或下移一行	4. 状态转换图
E. 多个相关研究领域的计算机专家	5. 图标菜单
F. Windows 屏幕采用了艺术化的 Web 类型的“界面”	6. 屏幕颠簸
G. 在一个窗口中的独立的区域	7. 有助于记忆的语法
H. 与某个特定的 OS 无关的用户界面	8. B2C

- 11. 为什么在应用中使用笔？
- 12. 窗口和框体是什么关系？
- 13. 弹出菜单有什么特征？
- 14. 用户界面设计过程包括哪些步骤？
- 15. 使用什么工具可用于提高对话框的吸引力？

(续)

I. 用来显示屏幕变化和顺序的图形工具	9. 对话
J. 在窗口中显示大型主机屏幕格式的软件	10. 滚动
K. 对用户有意义的命令语言界面	11. 情报学
L. 基于 B2C 事务的功能	12. 分页
M. 在主窗体中的菜单属性的图形化表示	13. 框体

- 5. 曾经，大多数软件都附带一本厚厚的用户手册。由于复杂的联机帮助系统和教材的流行，这些用户手册中许多已经消失了。如果你是一位为某个新应用软件设计帮助系统的系统设计人员，需要考虑记住哪些重要的因素？
- 6. 对于一个应用软件来说，在其用户界面中使用上百个屏幕、窗口和对话框并不是不常见。协调这些要素显示的顺序和条件可能是一个困难的过程，而且容易出错。为了辅助协调和记录这个过程，使用状态转换图显示屏幕、窗口和对话框显示的条件和顺序。

在你所在企业或学校中选取一个你熟悉的应用软件。为系统的一部分创建一个状态转换图，使用图 16-16 作为例子。
- 7. 假设你是一个项目团队的一部分，该团队受雇于一个正从大型主机技术向客户/服务器技术转移的公司。你正在进行用户界面设计。公司想把这个应用软件作为以后开发的软件的范本。你被授予自由支配开发这个软件和后续软件使用的用户界面屏幕规范和标准。创建一份一页纸的清单，列出你认为是最重要的规范和标准。
- 8. GUI 和 Web 应用软件为用户提供了使用应用软件的不同方式。提供这种用户友好性和灵活性的代价是复杂的设计和编程。拥有用户友好、灵活的界面而不需要复杂的程序设计是可能的吗？

项目和研究

1. 本书参考了另一个作者——Wilbert Galitz。Galitz 是一个现代的作家（另一些也在参考资料一节参考了），他被认为是人机界面领域的领袖人物。通过因特网查询最近关于人机界面设计和人机工程指南主题的文章和讨论。
 - a. 描述你找到的文章，包括作者和作者的观点。
 - b. 讨论和比较你发现的关于这个主题的任何不同的观点。
 - c. 作者们对于 Windows 和浏览器界面合并的趋势是什么看法？他们认为最终二者之间将只有很小的差异吗？
 - d. 对于未来可能在根本上改变人机界面设计的技术革新，他们有什么预测？
 - e. 基于你的研究，你认为在人机工程和人机界面设计领域的研究是它将要显现的那样复杂和先进吗？为什么？
2. 自动化的屏幕设计工具越来越强大和复杂。通过网络寻找几个主流的设计工具。访问它们的厂商的网站，并了解它们的特征。如果可以得到试用版，下载它们。
 - a. 你找到了什么自动化的屏幕设计工具？谁生产它们？
 - b. 比较它们的特点和功能。在一个矩阵中描述它们的不同特征。
 - c. 如果你是一位独立的设计人员，你会选用哪种工具？为什么？
 - d. 通过使用某一个工具，在你的产量方面你期望会看到明显的差别吗？有多大差别？
 - e. 你觉得使用这些工具会提高还是会限制你的创造力？解释你的回答。
3. 经常可以听到谈论“重新设计”政府部门使其运行得更像私人企业。这引发了是否公共领域和私人领域之间存在根本性区别的问题，是否这种差别会影响到系统？应该根据系统是用于政府部门还是用于私人企业采取不同的设计。
 - a. 调查私人领域和公共领域的系统设计人员。询问他们在设计人机界面时最主要的问题是什么？
 - b. 你发现有什么差别？
 - c. 有什么相似之处？
 - d. 你认为存在足够的共性，使得同样一套设计指南既可以用于公共领域，也可以用于私人领域吗？为什么？
 - e. 假设工资和收益相同，你更愿意作为公共部门的系统设计人员？还是私人企业的系统设计人员？为什么？
4. 为 B2C 和 B2B 网站设计界面屏幕被认为同其他类型的界面屏幕设计相比存在根本的不同。特别是，这些网站的目的是吸引消费者和企业购买它们的产品或服务。
 - a. 研究这个主题的文章。你找到什么文章，它们持什么观点？
 - b. （如果存在的话）总结在设计传统的输入/输出人机界面屏幕和设计 B2C 和 B2B 网站屏幕之间的不同？
 - c. 你同意这些观点吗？为什么？
 - d. 你认为哪种类型的背景对于 B2C 和 B2B 网站更有价值——系统设计方面的背景，还是市场和广告方面的背景？解释你的回答。

小型案例

有一个关于在界面设计中的人的因素的人机工程因素的讨论。讨论的主旨是我们了解将使用系统的用户，以及我们创建一个用户理解并能使用的系统界面是十分必要的。但这并不是一个学术问题；它是一个人的技能、人的理解问题。

1. 同你熟悉的完全非技术人员交谈。你的目标是理解那个人以及他的计算机需求和期望。你需要在交谈中考虑以下事情：
 - a. 作为一个普通人理解那个人：他们是谁？他们喜欢什么，不喜欢什么？他们有兄弟吗？有小孩吗？喜欢什么运动？什么爱好？他们在家里工作还是在外工作？如果你在他們自己的

“空间”（家里、办公室等）交谈，注意他们的个人影响。这些事情告诉你关于这个人些什么？

- b. 作为一个计算机用户理解那个人：他们有哪些计算机方面的经验？他们使用计算机做哪类事情？他们将不使用计算机做什么？有什么事情他们认为计算机特别有用吗？有什么事情特别迷惑？
- c. 当你询问这些问题时，他们的肢体语言告诉了你什么？他们对你感觉轻松吗？注意他们对你的反应，你如何着装，你所说的，以及你自己的肢体语言。

2. 使用你在小型案例 1 中获得的信息，为你交谈的人设计一个界面。你正在进行哪些界面设计修改，以使程序能够适应这个人？详细解释，并把结果提交老师。
3. 会见你在小型案例 1 中交谈的人，并向他们展示你创建的设计原型。获得他们关于设计的反馈。他们喜欢这个设计吗？他们能够在页面之间浏览吗？输入的设计如何？他们要修改什么吗？他们特别喜欢和不喜欢界面哪些内容？再一次，观察

团队和个人练习

1. 团队或个人练习：写下当你描述一个系统时所有使用的常见的计算机/技术行话。作为一个团队或个人，写下一份你的技术词汇清单的非技术的（“任何人都可以理解的”）版本。在课堂上进行圆桌讨论。
2. 团队练习：考虑电视机。什么使其如此容易使用？（几乎每个人都会看电视）如果需要重新设计，你要重新设计什么，如何重新设计？
3. 个人练习：反省你自己的优缺点。（a）花些时间谈论一些你自己的优点。（b）确定一件特别的活动能够揭示你的弱点，然后去做这个活动。
4. 根据你的第 2 次交谈（小型案例 3），修改你的界面设计。然后，以一种专业的完全可发布的方式提交你前面三个小型案例的工作和这个修改版本给老师。确保包含了一段关于你从你交谈的人和这次经历中学到了哪些知识的简短讨论。

使用 UML 进行面向对象设计和建模

本章概述和学习目标

本章是介绍系统开发中的面向对象工具和技术的一个章节中的第 2 章，重点介绍系统设计期间的面向对象建模工具和技术。本章将具体介绍以下内容：

- 区分实体类、接口类、控制类、持续类和系统类。
- 理解依赖关系和导航关系的概念。
- 定义可见性和它的三个层次。
- 理解对象责任的基本概念，以及它与对象之间发送消息的关系。
- 描述面向对象设计包括的活动。
- 区别设计用例描述和分析用例描述。
- 描述 CRC 卡建模。
- 使用顺序图建模类交互。
- 构造反映设计说明的类图。
- 使用状态机图建模对象状态。

本章关键术语

面向对象设计（Object-Oriented Design, OOD）采用协作的对象、对象的属性和方法说明软件解决方案的一种方式。

实体类（entity class）是承载业务相关信息并实现分析类的对象类。

接口类（interface class）是提供参与者与系统交互方式的对象类。例如：窗口、对话框或者屏幕。对于非人类参与者，应用编程接口（API）就是接口类，有时称为边界类。

控制类（control class）是承载了应用逻辑的对象类。这种逻辑如：涉及多个实体对象类的业务规则和计算。控制类协调接口类和实体类之间的消息，以及消息发送的顺序。

持续类（persistence class）是提供读写数据库中的持续属性的功能的对象类。

系统类（system class）是处理操作系统相关功能的对象类。

可见性（visibility）是外部对象对某个属性或方法的访问等级。

方法（method）是响应一条消息而执行的软件逻辑。

对象责任（object responsibility）是当被请求时对象必须提供服务的义务，如果需要，它应该与其他对象协作以满足请求。

顺序图（sequence diagram）是一种 UML 图，它通过描述对象按照时间顺序的消息交互来建模用例逻辑。

对象状态（object state）是对象在其生命周期中某一点所处的条件。

状态转换事件（state transition event）通过修改对象属性的一个或多个值触发对象状态变化的事件。

状态机图（state machine diagram）是一种 UML 图，它描述了一个对象在其生命周期中可能的状态组合，触发状态转换的事件，以及决定状态转换的规则。也称状态表图或状态转换图。

角色扮演（role playing）是通过扮演一个对象的行为和责任模拟对象行为与协作的行动。

设计类图（design class diagram）是一个图形，描述了与用于构建程序的软件组件相对应的类。

17.1 设计面向对象系统

在第 9 章中，我们学习了对象类。那么这些类如何组合在一起构成应用系统呢？面向对象系统是

什么样子？在一个纯面向对象环境中，每一段代码都存在于某个对象类内部——所有的用户界面，所有的程序逻辑，等等。应用系统通过让类发送消息并从其他类接收消息而工作。面向对象设计（OOD）的目标是说明系统的对象和消息。

图 17-1 展示了用 C#. NET 创建的 Web 页面的程序代码。这个 Web 页面提供了 SoundStage 系统的部分用户界面（注意文本框、标签和按钮）。在代码的开始，我们看到“public class Login”，这指出了所有的用户界面代码存在于某个类内部。在屏幕的底部，这个用户界面类创建了一个成员类的实例，并调用了它的 validateLogin 行为（方法）。一个面向对象系统至少被结构化为三种不同类型的对象类。



图 17-1 一个面向对象应用

17.1.1 实体类

实体类通常对应现实生活中的实体（例如成员或者订单），它包含了用于描述实体的不同实例的信息（称为属性），还封装了维护其信息或属性的行为（称为方法）。这些是我们在第 9 章定义的对象类的种类。它们是系统的核心。

17.1.2 接口类

用户通过接口类实现的用户界面同系统通信。描述用户直接同系统交互的用例功能应该放在接口类中。接口类的责任包括以下两方面：

- 1. 将用户的输入翻译成为系统可以理解和应用以处理业务事件的信息。
- 2. 获取关于业务事件的数据，并把数据加以翻译，提交给用户。

每个参与者或用户都需要通过各自的接口类同系统通信。在某些情况下，用户可能需要多个接口类。以 ATM 机为例，它不仅需要一个显示器用于显示信息，而且还需要读卡器、现金分配器和收据打印机，所有这些都将被看作接口对象类。

17.1.3 控制类

控制类实现系统的业务逻辑或业务规则。一般来说，每个用例由一个或多个控制类实现。控制类通过向实体类发送消息和从实体类接收消息，处理来自接口类的消息并响应这些消息。

一个面向对象系统可以只用这三种类实现。但许多方法学还包含另外两种类。

17.1.4 持续类

实体类的属性通常是持续性的，这意味着它们超出系统的运行而继续存在。读写数据库中的属性

的功能将在实体类中构建。但如果这项功能被放到独立的持续类（或数据访问类）中，实体类将保持实现中立。这会使实体类具有更好的复用性，这是面向对象设计的一个主要目标。

17.1.5 系统类

最后一种对象类——系统类将其他对象从操作系统相关功能独立分开。如果系统被移植到另一个操作系统，只有这些类以及一些接口类可能需要修改。

为什么有这些种类的类？这样构造系统使得维护和增强那些类变得更容易、更简单。

17.1.6 设计关系

在面向对象分析中，我们重点关注最常见的对象关系：关联关系、聚合关系和泛化/特化关系。在面向对象设计中，需要建模更深层的关系，以便正确地说明软件组件。下面就介绍这些关系。

17.1.6.1 依赖关系

依赖关系用于描述下面两种情况下两个类之间的关联关系：1) 指出当一个变化出现在一个类中，它可能会影响到另一个类；2) 支持一个持久类和一个临时类之间的关系。接口类一般是临时的，可以使用这种方式建模。请注意图 17-2。在这个例子中，ORDER DISPLAY WINDOW 类是一个接口类，被创建用于显示订单内容。它依赖于 PLACE NEW ORDER HANDLER 类，将订单信息映射过来，并响应接口发起的事件。依赖关系使用虚线箭头表示。

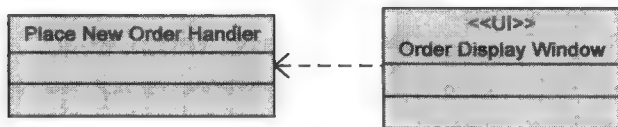


图 17-2 依赖关系举例

17.1.6.2 导航能力

如第 9 章所述，类之间的关联关系默认的是双向的，意味着一类对象可以导航到（发送消息）另一种类。也有很多情况下，你希望限制消息仅向一个方向发送。例如，假定每个系统用户必须有一个口令，而且用户必须每隔 30 天换一次口令。我们还假设当用户修改口令时，新口令不能是他在过去 6 个月中使用过的口令。这个场景的模型描述在图 17-3 中。给定一个 USER，你要找到用户的当前 PASSWORD，用于认证或者修改口令。因此，USER 类要发送一条消息给 PASSWORD 类。在大多数情况下，给定一个 PASSWORD，确定对应的 USER 是没有意义的。导航能力采用指向消息发送方向的箭头表示。



图 17-3 导航能力举例

17.1.7 属性和方法可见性

可见性定义了属性和方法如何被其他对象访问。UML 提供了 3 个层次的可见性：

1. 公共——用符号“+”表示
2. 保护——用符号“#”表示
3. 私有——用符号“-”表示

公共属性可以被其他任何对象（或类）的任何方法访问，公共方法可以被任何其他对象（或类）的任何方法调用。定义保护属性（方法）的类或其子类可以访问该属性（方法）。私有属性（方法）只能由定义它的类访问。如果需要调用某个方法以响应另一个对象发送的消息，该方法应该声明为公共的。在大多数情况下，所有的属性都应该声明成私有的，以强制封装。图 17-4 是一个表示属性和方法可见性的例子。

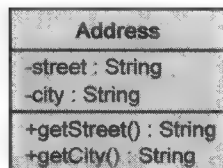


图 17-4 可见性的例子

17.1.8 对象责任

在面向对象系统中，对象封装了数据和行为。在设计中，我们集中确定一个系统必须支持的行为，然后，设计实现这些行为的方法。通过这些行为，确定一个对象责任。

在第 9 章中，我们知道了对象具有行为，或者它们能够做的事情。在面向对象设计中，认识到对

象具有责任也很重要。对象责任同对象能够发送和/或响应消息的概念密切相关。如图 17-5 所示，一个订单类具有显示一个客户的订单的责任，但它可能需要同客户（CUSTOMER）类协作以获取客户数据，其次同会员订单产品（MEMBER ORDERED PRODUCT）类协作以获取订购的产品数据。会员订单产品类无法自行完成全部请求，所以它需要同产品（PRODUCT）类协作，获得产品的详细信息。因此，当每个类收到请求服务的消息时，它有责任响应消息并实现请求。

类责任与类方法不同。类责任通过创建一个或者多个方法实现，这些方法可能同其他对象或方法协作，如上所述。

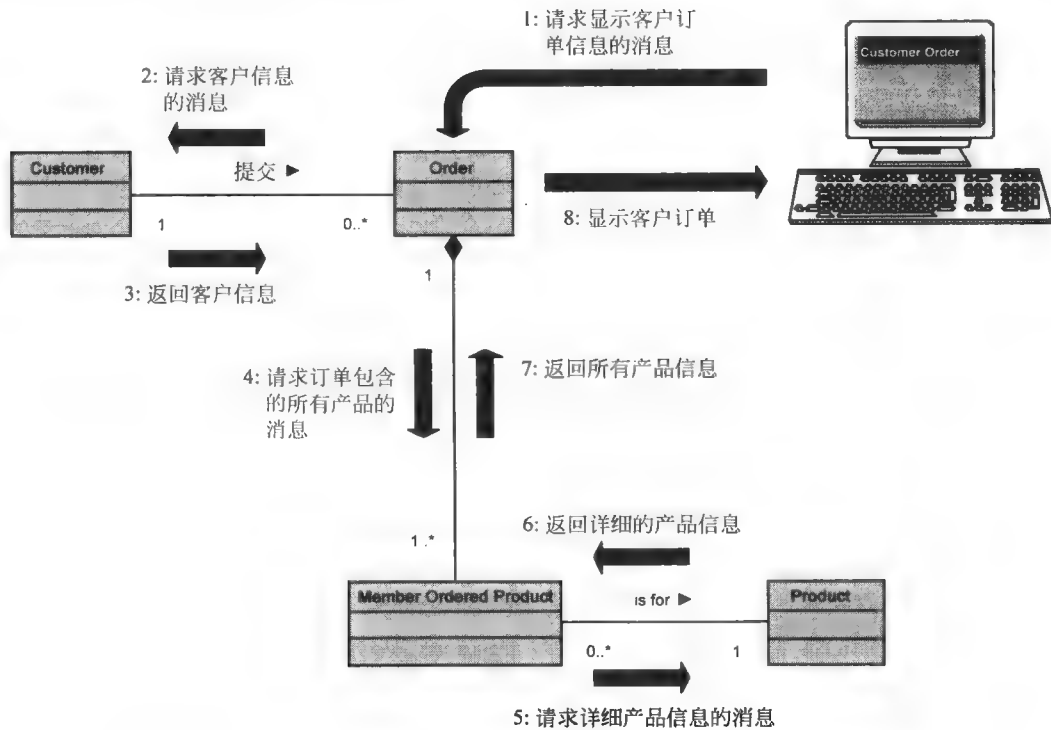


图 17-5 对象责任

17.2 面向对象设计过程

在面向对象分析（OOA）中，我们基于理想条件并依赖一些硬件或软件方案确定对象和用例。在面向对象设计（OOD）期间，我们则要对这些对象和用例加以精炼，以反映所建议的方案的实际环境。

面向对象设计包括以下活动：

1. 对用例模型加以精炼以反映实现环境。
2. 建模支持用例情境的对象交互、行为和状态。
3. 修改对象模型以反映实现环境。

在以下的小节中，将介绍这些活动，并学习面向对象设计的步骤、工具和技术。

17.2.1 精炼用例模型

在用例建模的这次迭代中，我们将对用例加以精炼，从而包括参与者（或用户）如何实际地与系统接口以及系统如何响应刺激处理业务事件的细节。用户访问系统的方式——通过菜单、窗口、按钮、条形码阅读器、打印机等——都应该被详细地描述，窗口、报告和查询的内容也应该在用例中加以说明。虽然精炼用例经常是耗时而且繁琐的，但是这项工作必须完成。这些用例将是系统实现期间开发用户手册和测试脚本的基础。而且，在系统实现期间，程序员将使用这些用例构造应用程序。

在下面的步骤中，我们将调整每个用例，使它们适应实现环境或“现实”，并且记录下调整的结果。重要的是每个用例都被高度精炼地细化，并且描述了用户同系统的交互。用户可以使用这些精炼后的用例验证系统设计，并且程序员也可以使用用例说明过程和接口。

17.2.1.1 第 1 步：将“分析”用例转换成“设计”用例

在第 6 章和第 12 章中，我们学到了如何在系统分析期间进行用例建模，记录用户对一个给定业务情境的需求。在这一步中，我们精炼那些用例，使用用例能够反映新系统实现环境的实际情况。

图 17-6 演示了对系统分析期间定义的下新订单用例的精炼。这个版本被标记为设计用例，以便同以前完成的分析版本相区别。我们希望精炼后的设计用例与原来的分析用例相独立，从而获得为各种不同物理实现复用用例的最大灵活性。请注意图 17-6 中的以下注释：

- ①用例类型——反映实现细节，例如：用户界面约束条件，称为系统设计用例的战术用例从系统分析用例中导出。
- ②窗口控制——在系统设计用例中，显式陈述的窗口控制，例如：图标、链接、复选框和按钮。
- ③窗口名称——每个用户界面元素的名称（窗口名称）。如果存在关于用户界面元素的额外信息，也要索引它们。否则，更详细的窗口说明将被添加到用例中。
- ④导航指令——需要说明关于用户如何在用户界面中导航的指示。

17.2.1.2 第 2 步：修改用例模型图和其他文档以反映新用例

所有的系统分析用例都被转换成系统设计用例后，仍可能会发现新的用例、用例依赖，甚至参与者，所以保持文档的正确性和时效性十分重要。因此，在这一步中，应该修改用例模型图、用例依赖图、参与者和用例词汇，以反映第 1 步中引入的新信息。

17.2.2 建模支持用例情境的类交互、行为和状态

17.2.2.1 第 1 步：确定并分类用例设计类

在上一节中，我们精炼了用例以反映实现环境。在这个活动中，我们将确定并分类设计类，这些设计类说明了用例中的功能需求，而且还要确定类之间的交互、类责任和行为，见图 17-7。

- ①接口类列包括了用例中提到的与用户直接接口的对象，例如屏幕、窗口、读卡器和打印机。参与者或用户同一个系统接口的唯一方式是通过一个接口类，所以每个参与者或用户至少应该有一个接口类。
- ②控制类列包括了封装应用逻辑或业务规则的类。每个用例应该为用户或参与者揭示一个控制类。
- ③实体类列包括了其属性在用例中被引用的业务领域对象。

17.2.2.2 第 2 步：确定类属性

在分析和设计期间，都可以确定类属性。在将分析用例转变成为设计用例的工作中，我们开始引用用例文本中的属性。在这一步中，我们检查每个用例中以前没有被确定的附加属性，而且还要修改类图以包括这些属性。

17.2.2.3 第 3 步：确定类行为和责任

一旦确定了支持用例功能所需的所有对象，我们重点定义它们的特殊行为和责任，这一步包括以下任务：

- 分析用例以确定所需的系统行为。
- 关联行为和责任到类。
- 建模具有复杂行为的类。
- 检查类模型附加行为。
- 验证分类。

在第 9 章，我们了解到类封装了数据和行为。在确定类行为和责任上的第一个任务再次通过检查用例实现。我们检查用例描述，以确定所有的动词短语。动词短语建议了完成一个用例情境所需的行为，这些动词短语同响应一个俱乐部会员提交一个新订单的业务事件所需的行为相关。应该分别检查每个用例，并确定同用例相关的行为。

会员服务系统

作者: K. Dittman

日期: 11/21/06

版本: 1.00

用例名称	下新订单	①用例类型 业务需求：□ 系统分析：□ 系统设计：☑
用例 ID	MSS-SUC002.00	
优先权	高	
来源	需求——MSS-R1.00 需求用例——MSS-BUC002.00	
主要业务参与者	俱乐部会员（别名——活动会员、会员）	
主要系统参与者	俱乐部会员（别名——活动会员、会员）	
其他参与者	● 仓库（别名——分销中心）（外部接收者） ● 应付账/应收账（外部服务者）	
其他有兴趣的关联人员	● 市场部——对销售活动感兴趣，以计划新的促销 ● 采购部——对销售活动感兴趣，以补充库存 ● 管理层——对销售活动感兴趣，以评估公司性能和顾客满意度	
描述	该用例描述一个俱乐部会员通过因特网提交一个音阶娱乐俱乐部产品的订单。会员选择他想购买的项目。一旦会员完成了采购，会员的资料信息以及他的账号被验证。一旦验证产品有库存，就向仓库发出一个发货订单准备发货。对于没有库存的产品，生成一个退单。一旦完成，会员将得到一份订单确认。	
前提条件	个人必须是注册的系统用户。 会员必须已经登录到系统中，会员的个人主页已经显示。	
触发器	当会员选择输入新订单时，用例被触发。	
典型事件过程	参与者动作	系统响应
	<p>第 1 步：会员单击下新订单图标（或链接）。</p> <p>●</p> <p>第 3 步：会员使用滚动条按钮、[Page Up] 按钮和 [Page Down] 按钮浏览可得到的条目，或者使用第 2 步提供的导航控制。会员通过单击复选框选择他想购买的项，并输入订购数量。</p> <p>①</p> <p>第 5 步：会员验证个人信息（发货和收费地址）。如果没有变化，会员单击 [Continue] 按钮。</p>	<p>第 2 步：系统做出响应，显示窗口 W11：目录显示，列出音阶俱乐部产品清单。*</p> <p>●</p> <p>如果产品清单大于 50（显示在一页上的最大数量），系统计算显示产品所需的页数。然后系统为会员提供所需的导航按钮，例如：[第 1 页]、[前 1 页]、[下 1 页]、[最后 1 页] 和 [1] [2] [3] [4]，等等。</p> <p>第 4 步：一旦会员完成了选择，系统访问会员的个人信息（发货和收费地址），在窗口 W02：会员资料显示中显示。系统还提示会员进行所需的修改。</p> <p>第 6 步：对于订购的每个产品，系统验证产品可用性，决定发货日期，决定向俱乐部会员收取的价格，决定订单的总价格。如果某项不能马上得到，指出产品退单，或者还没有发货（对于预订）。如果某项不再可得到，也需要指出。系统然后在窗口 W03：订单总结显示中给会员显示一个订单总结供确认。系统还提示会员进行所需的修改。</p>

* 参考窗口内容和规格的用户界面说明。

图 17-6 设计用例举例

一旦确定了行为, 我们就必须确定行为是手工的还是自动的。如果是自动的, 它们就必须同对执行那个行为负有责任的相应对象类型相关联。图 17-8 列出了“下新订单”用例的每个动词词组或行为, 是手工的还是自动的, 以及要关联的对象类型。

在图 17-9 中, 我们精简了行为列表, 只显示需要自动化的行为。注意对象类型在前面 (第 1 步) 已经定义了。在下一个任务中, 将使用这个列表分配对象行为。

接口类	控制类	实体类
W00-Member Home Page	Place New Order Handler	Billing Address
W02-Member Profile Display		Shipping Address
W03-Display Order Summary		Email Address
W04-Display Order Confirmation		Active Member
W09-Member Account Status Display		Member Order
W11-Catalog Display		Member Ordered Product
W15-Product Detail Display		Product
		Title
		Audio Title
		Game Title
		Video Title
		Transaction

图 17-7 “下新订单”用例的接口类、控制类和实体类

行 为	自动/手工	对象类型
Process new member order	手工/自动	控制
Click icon to place new order	手工	
Retrieve Product catalog information	自动	实体
Display W11-Catalog Display Window	自动	接口
Scroll or page through catalog	手工	
Select product to be ordered and enter quantity	手工	
Retrieve member demographic information	自动	实体
Display W02-Member profile Display window	自动	接口
Verify member demographic information	手工	
Validate quantity amount	自动	实体
Verify the product availability	自动	实体
Determines an expected ship date	自动	实体
Determine price of product	自动	实体
Determine cost of the total order	自动	实体
Display W03-Order Summary Display window	自动	接口
Prompt user	自动	接口
Verify order information	手工	
Check Status of member account	自动	实体
Prompt user for payment option	自动	接口
Store order information	自动	实体
Record back order information	自动	实体
Generate order confirmation	自动	实体
Display W04-Order Confirmation Display	自动	接口
Click button or icon	手工	

图 17-8 “下新订单”用例行为总结

行 为	对象类型
Process new member order	控制
Retrieve Product catalog information	实体
Display W11-Catalog Display window	接口
Retrieve member demographic information	实体
Display W02-Member Profile Display Window	接口
Validate quantity amount	实体
Verify the product availability	实体
Determines an expected ship date	实体
Determine price of product	实体
Determine cost of the total order	实体
Display W03-Order Summary Display window	接口
Prompt user	接口
Check Status of member account	实体
Prompt user for payment option	接口
Store order information	实体
Record back order information	实体
Generate order conformation	实体
Display W04-Order Confirmation Display	接口

图 17-9 “下新订单”用例精简的行为列表

下一个任务是确定与一个类型关联的所有行为，并确定类之间的协作。记录对象行为和协作的一种流行工具是类责任协作（CRC）卡^①。图 17-10 描述了 MEMBER ORDER 类的 CRC 卡，其中包含了与该对象类关联的所有用例行为和责任。

对象名称: Member Order	
子对象:	
超对象: Transaction	
行为和责任	协 作 者
Report order information Calculate subtotal cost Calculate total order cost Update order status Create Ordered Product Delete Ordered Product	Member Ordered Product

图 17-10 MEMBER ORDER 类的 CRC 卡

CRC 卡可以由一个交互过程开发和细化，其中卡片在一组系统分析员或用户之间分配。然后他们经过用例场景的步骤，使用海绵球扮演所需的协作。主持人开始将球传给持有最初响应场景的类的卡片的人员。这个人描述实现该责任所需的逻辑。如果该类需要它没有的信息，或者必须修改它没有的信息，那么这个人就将球传给具有那个信息的卡片的人员。传球表示了两个类之间一种所需的协作，将记录在卡片上。当场景扮演完成之后，球被送回主持人。

对用例情境的分析可能没有揭示给定对象类型的所有行为。另一方面，通过检查类图，你可能会发现需要分配给定对象类的没有在用例情境中提到的其他行为。例如：分析图 17-12 中的类之间关系，如何创建或者删除这些关系？哪个责任应该分配给哪个对象？一般的规则是，控制关系的类应该负责

① CRC 卡是由 Kent Beck 和 Ward Cunningham 提出来的。

创建和删除关系。注意图 17-12 中 MEMBER ORDER 和 MEMBER ORDERED PRODUCT 之间的关系。系统通过指定 MEMBER ORDER 类具有一个“增加订购的产品”的行为，有效地赋予了 MEMBER ORDER 类创建关系的能力。而且，第 9 章中介绍了 4 种“隐含”关系可以关联到任何对象类上，即：创建新实例、修改数据或属性、删除实例和显示关于对象类的信息。当检查用例以确定并关联对象类和行为时，我们还要专门确定对象类之间所需的协作或协同。在图 17-10 中，MEMBER ORDER 类需要同 MEMBER ORDERED PRODUCT 类的协作以访问关于每个被订购产品的信息。如果一个类需要另一个类的属性以实现一个行为，协作类就需要具有一个行为或方法以提供那个属性。

确定对象类型之间的协作是必需的，以确保所有的用例类协调地工作，完成触发用例场景的业务事件所需的过程。

另一种发现和记录类行为和责任的工具是顺序图。在第 9 章中，我们了解了系统顺序图，它是一种高层视图，描述了一个用例场景参与者和系统之间的交互。一个完整的顺序图描述了场景中涉及的所有对象类之间的交互。顺序图通过描述按照时间顺序对象之间的消息交互建模了用例（或用例的一部分）的逻辑。这些消息按照时间顺序自顶向下排列。

顺序图可以看作是一种综合了用例步骤和类图对象的方法。它可以用作与程序员交流的工具，用来说明在实现用例时调用什么方法（行为）。图 17-11 显示了图 17-6 中描述的“下新订单”用例的基本步骤 6 的一个场景。图 17-11 中展示了以下的顺序图记号：

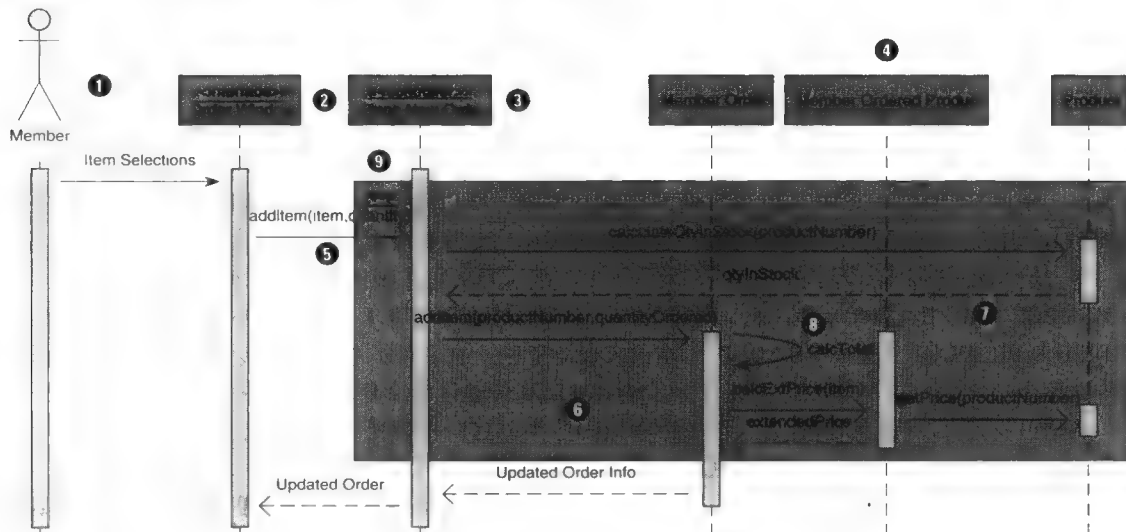


图 17-11 “下新订单”用例的第 6 步的顺序图

- ① 参与者——同用户界面交互的参与者使用用例角色符号表示。有时为了简化而省略了参与者。有时参与者使用一个类似类的带记号 << actor >> 的框表示。从参与者向下伸展的垂直虚线表示了顺序的生命。
- ② 接口类——框代表用户界面类代码。为了确保不会混淆这是哪一种对象类，加注 << interface >>。与 UML 中的其他记号一样，最利于沟通的就是正确的。(:) 是标准的顺序图记号，表示类的一个运行实例。从类向下伸展的垂直虚线表示了顺序的生命。
- ③ 控制类——每个用例都将会有一个或多个控制类，采用与接口类同样的绘制方法，加注 << controller >>。
- ④ 实体类——为每个需要在步骤序列中协作的实体类添加一个框。冒号 (:) 表示了对象实例，即一个特定的订单、产品等。
- ⑤ 消息——水平实线箭头表示了发送到类的消息输入。每条消息调用箭头所指向的类的行为（或方法）。消息的 UML 记法是第 1 个单词小写，后续的单字第 1 个字母大写，单词之间没有空格。

在圆括号中,包含了需要传递的参数,遵循同样的命名规范,用逗号分隔每个参数。

- ④ **活动条**——设置在生命线上的框条表示了每个对象实例存在的时间周期。如果你熟悉某种面向对象程序设计语言,你应该记得在程序中需要实例化对象使其工作。活动条指示了在内存中一个实例的生命周期。一般来说,对象为了响应消息而被实例化。当然,持续对象将会作为存储的数据一直存在。
- ⑤ **返回消息**——水平虚线箭头是返回消息。每个行为都应该返回点什么,至少一个真/假消息,表示行为是否成功。但为了简单起见,经常假定存在返回消息,而在顺序图中省略。
- ⑥ **自我调用**——对象可以调用它自己的方法。
- ⑦ **框架**——在第9章中,我们看到了在系统顺序图中如何使用框架来表示一条或多条消息是可选的步骤。这里我们使用框架来表示控制需要循环通过其中所有的要素。

让我们浏览一下图 17-11 中显示的顺序图。会员使用 ORDER WINDOW (被标示为一个接口类) 中提供的屏幕工具做出其选择。然后 ORDER WINDOW 传递这些包含项目和数量说明的选择给控制类。CONTROLLER 循环遍历每个项目。用例要求对于每个订单项目,系统必须验证产品的可用性。为此,CONTROLLER 发送消息给 PRODUCT,调用它的 calculateQtyInStock 方法。我们已经确定了 calculateQtyInStock 方法是 PRODUCT 的一个行为,所以可以从类图中找到它,并把它拖到这里。如果它不是一个已经存在的行为,那么可以从这个顺序图中确定它需要存在,然后把它添加到类图中。为什么这个行为被指定给 PRODUCT? 从图 17-11 中,我们看到 PRODUCT 有一个 quantityInStock 属性,所以这是自然的信息来源。PRODUCT 返回 quantityInStock 给 CONTROLLER。用例中包含了处理不在库存的商品的空话,但我们不用遵循那个场景。这个顺序图假定所有的项目都在库存中。

每个库存中的项目必须被添加到订单中。这应该是 MEMBER ORDER 或 MEMBER ORDERED PRODUCT 的责任吗? 从图 17-12 中看到, MEMBER ORDER 和 MEMBER ORDERED PRODUCT 有一个组合关系,使得 MEMBER ORDER 负责实例的创建和删除。所以我们让 CONTROLLER 传递这条消息给 MEMBER ORDER。当它添加一个项目时, MEMBER ORDER 需要重新计算其总数。所以它调用它自己的一个方法 (calcTotal)。为了进行这个计算,它需要新项目的可扩展价格 (数量时间价格),所以它调用 MEMBER ORDERED PRODUCT 的 calcExtPrice。计算需要 PRODUCT 持有的价格信息,所以 MEMBER ORDERED PRODUCT 创建一个 PRODUCT 实例来查询价格。然后可扩展价格被传回 MEMBER ORDER,它传递整个订单给 CONTROLLER。最后, CONTROLLER 传递订单给 ORDER WINDOW 进行显示。

从这里我们可以确定哪些行为应该被指派给哪些类,它们将接受和返回哪些参数。一旦行为被确定、记录和关联到特定的类,就可以更新类图以包含相应类中的行为。

在继续向前之前,让我们看看另一个顺序图。图 17-13 显示了抽象用例“通过关键字查询产品目录”的一个简单的用例图,这是“下新订单”用例的替代第 3 步。当会员选择了“关键字查询”选项,并输入一个关键词,界面传递请求给控制器。控制器调用 PRODUCT 的 reportProduct 方法,传递关键词。PRODUCT 返回匹配关键词的一组产品。如果包含了持续对象,将显示到数据库的数据读入状态。

下面是构造顺序图的一些有用的指南:

- 确定顺序图的范围。你也许希望描述整个用例场景,或者只是一个步骤。
- 绘制参与者和接口类,如果你的范围包括这些内容的话。
- 沿左手边列出用例步骤。
- 对控制器类及必须在顺序中协作的每个实体类,基于它拥有的属性或已经分配给它的行为绘制的框。
- 如果你的范围包括这些内容的话,为持续类和系统类绘制框。
- 绘制所需的消息,并把每条消息指到将实现响应消息的责任的类上。
- 添加活动条指示每个对象实例的生命期。
- 为清晰起见,添加所需的返回消息。
- 如果需要,为循环、可选步骤、替代步骤等添加框架。

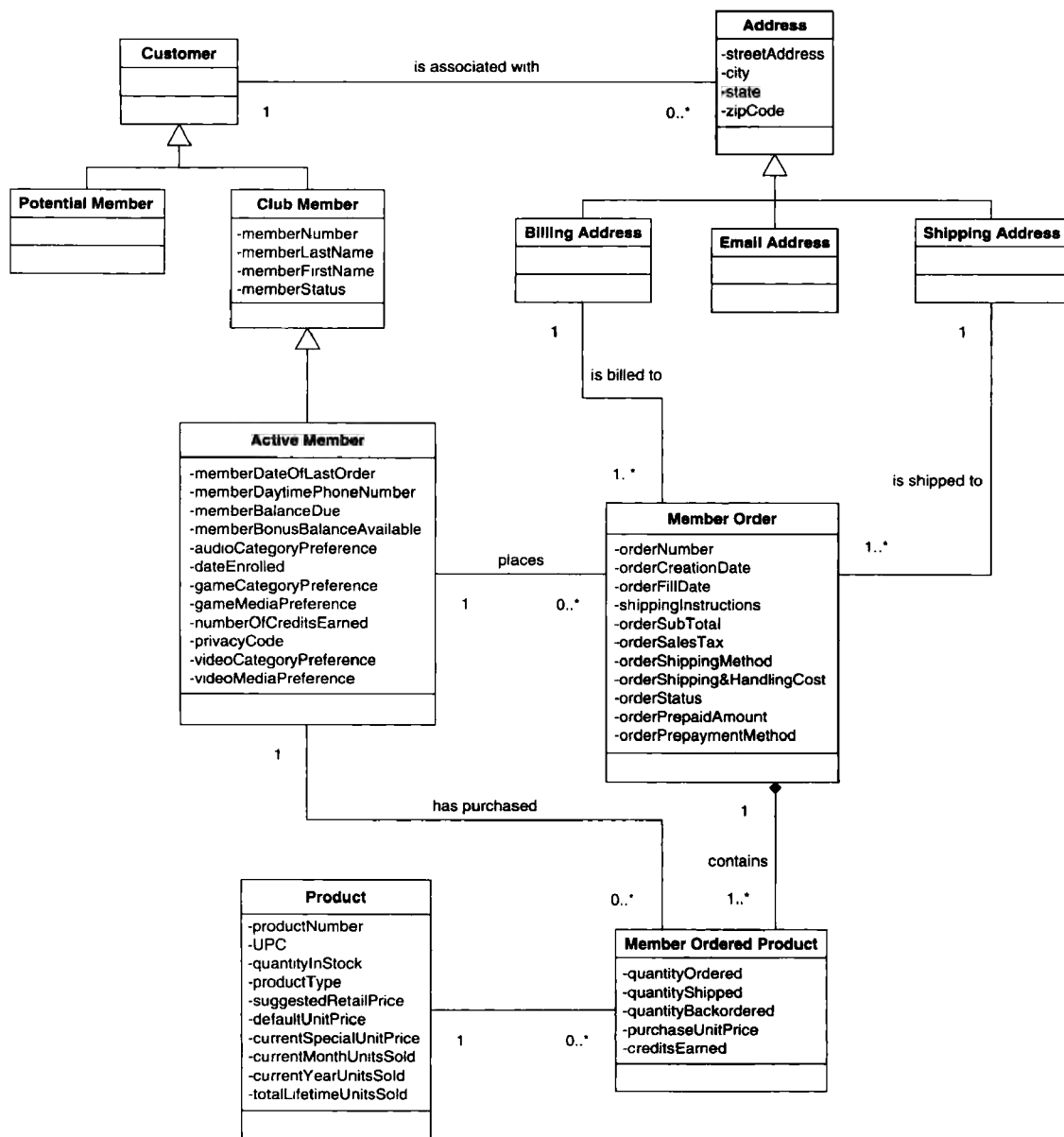


图 17-12 “下新订单”用例的部分类图

我们将在本章后面讨论设计模式时再次讨论顺序图。

17.2.2.4 第 4 步：建模对象状态

我们的下一个任务是基于对象的状态变化确定和建模具有复杂行为的对象。所有的对象都拥有状态——对象属性在某一个时刻的值。当对象的某个属性值发生变化时，对象状态发生变化。状态的变化由状态转换事件触发。图 17-14 显示了一个航天飞机停靠在发射架上（发射前状态），航天飞机起飞（一个事件）后，它改变状态（状态转换），当它在飞行中，处于飞行状态。我们还可以列出其他状态，例如：着陆、拆卸或者修理。业务系统中的许多对象具有复杂的行为，或者存在许多状态。

状态机图建模一个对象的生命周期。它描述了对象具有的不同状态，引起对象改变状态的事件，以及决定对象在状态间转换的规则。换句话说，它说明了对象可以从哪个状态转换到哪个状态以及在何种条件下转换。通过以下活动构造状态机图：

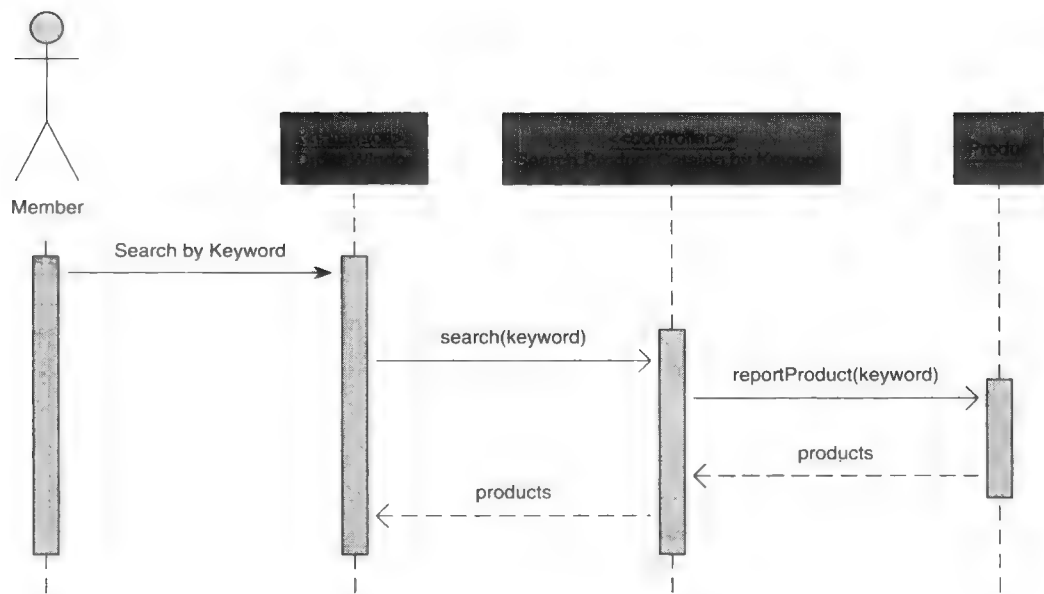


图 17-13 “通过关键字查询产品目录”的用例顺序图

- 确定初始状态和最终状态。（对象如何创建和销毁？）
- 确定对象在生命期中可能具有的其他状态。
- 确定引起对象离开某个特定状态的触发器（事件）。
- 确定状态转换路径。（当对象状态改变时，对象将进入的下一个状态是什么？）

图 17-15 是会员服务系统中 MEMBER ORDER 对象的状态图。它从一个初始状态（实心圆）开始，在不同状态（圆角矩形）之间转换，最后进入结束状态（内部空心的实心圆）。每个箭头表示了触发 MEMBER ORDER 对象从一个状态转换到另一个状态的事件。

不是所有的对象都需要状态机图。通常，只对那些明显具有可确认的状态和复杂行为的对象绘制状态机图。根据我们的经验，具有“status”属性的对象都是构造状态机图的好候选对象。

最后的任务是验证前面任务的结果，这包括与相应的用户一起进行走查。一种经常使用的验证方式是角色扮演。在角色扮演中，用例情境由参加者演出。参加者可以假扮参与者或对象类型的角色，合作处理一个假设的业务事件。消息发送可以使用一个东西模拟，例如一个在参与者之间传递（或者有时抛出）的球。角色扮演在发现遗漏的对象和行为以及验证对象间的协作方面十分有效。

17.2.3 修改对象模型以反映实现环境

一旦设计了对象及其所需的交互，就可以对类图加以精炼，以表示应用程序中的软件类。设计类图通常包括以下内容：

- 类
- 关联关系、泛化/特化关系、聚合关系
- 属性和属性类型信息
- 带参数的方法
- 导航能力

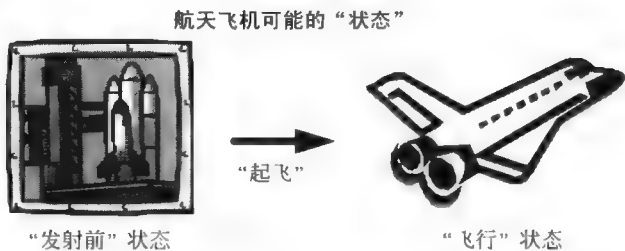


图 17-14 对象状态举例

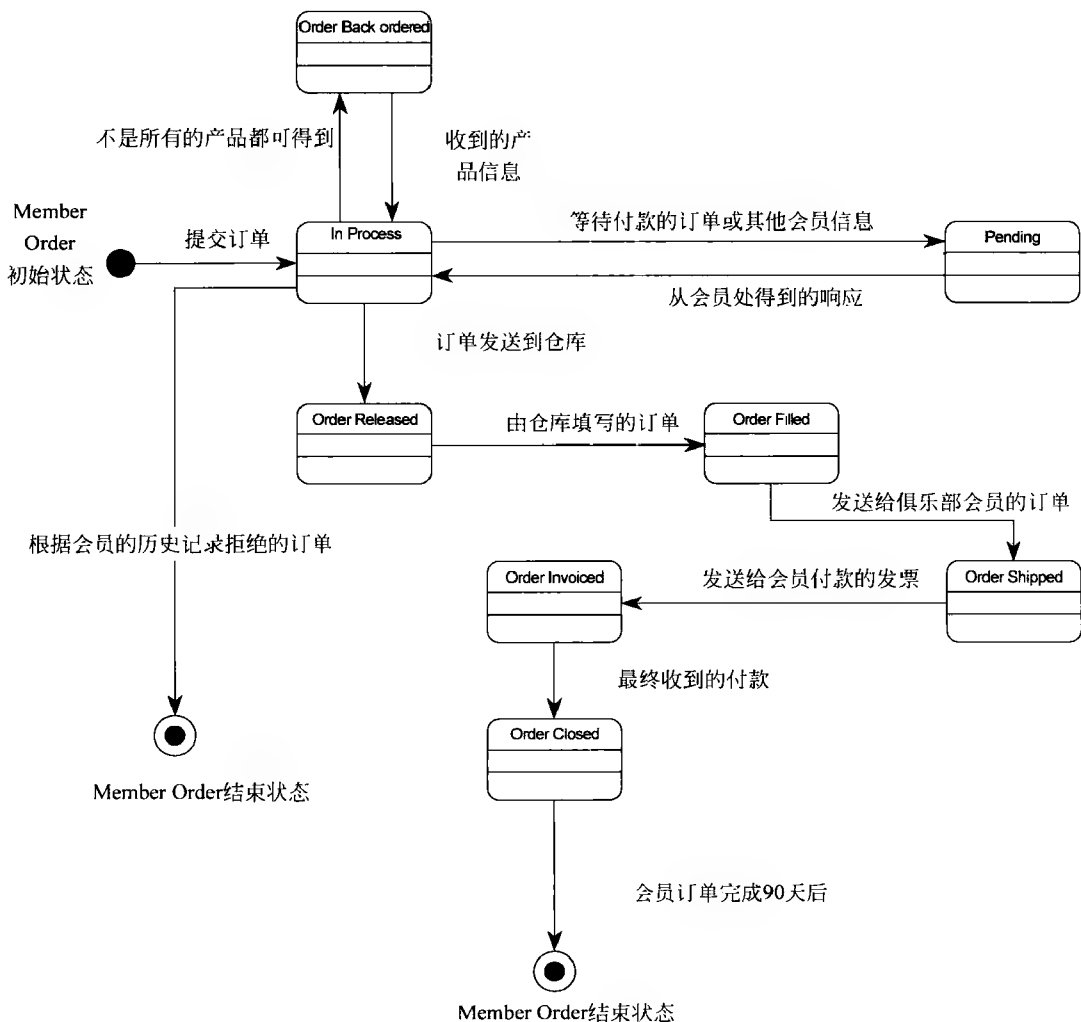


图 17-15 MEMBER ORDER 对象的状态图

• 依赖关系

以下步骤将 OOA 中准备的类图转换成设计类图：

1. 向图中添加设计对象。以前确定的实体、接口和控制对象应该添加到图中。出于图大小和可读性的考虑，应该只包含主要的接口对象。

2. 向设计对象中添加属性和属性类型。OO 编程语言允许使用通用的属性类型，例如：整数、日期、布尔值、字符串（文本）等。OO 语言也能够定义复杂的属性类型，例如：地址、社会安全号、电话号码，这对开发人员来说是一个强大的特征。

3. 添加属性可见性。属性可以定义为公共的、保护的和私有的。

4. 向设计对象添加方法。定义方法获取和修改每个对象的属性值。这类方法通常称为“setter”和“getter”方法。经常把这些方法从设计类图中去掉，以便节省空间，增加图形的可读性，因为它们总是默认地存在。另外，还包括实现前面确定的责任和行为的方法，例如：创建或删除类实例，或者形成或中断类关联。请注意方法名称基于所选的编程语言格式化。在 Smalltalk 中命名方法名称的方式与 Java 中不同。在本书中，我们将使用标准的 UML 格式：方法名称（参数列表）。

5. 添加方法可见性。方法可以定义为公共的、保护的和私有的。

6. 添加类之间的关联关系导航能力。给单向关联添加导航箭头，指示在源类和目的类之间发送有

方向性的消息。

7. 添加依赖关系。对于图中出现的用户界面类，绘制它们与控制对象之间的依赖关系线。

图 17-16 是音阶公司会员服务系统类图的部分视图，请注意以下内容：

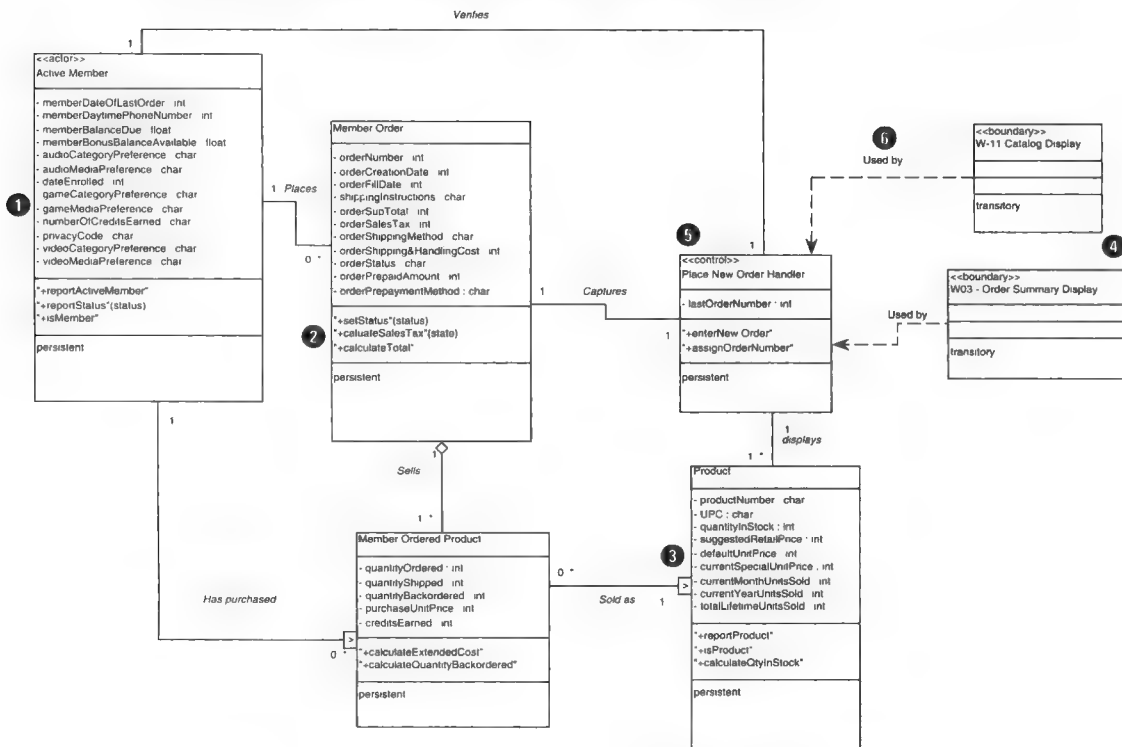


图 17-16 “下新订单”用例的部分设计类图

- ❶ 对每个属性指定了可见性。在这个特殊的例子中，所有的属性都是私有的，采用符号“-”表示。
- ❷ 指定了方法及其可见性。
- ❸ 一些关联上注明了导航能力，指出消息仅仅单向传送。
- ❹ 添加接口类显示主要的用户界面对象。在这个特殊的软件包中，它们被看作是边沿对象，本身是临时的。
- ❺ 添加控制类协调接口对象和实体对象之间的交互。我们也给这个控制类赋予了（顺序）分配新订单号的责任。
- ❻ 接口对象依赖于控制对象。

复习题

1. 在面向对象设计中使用哪三种对象？
2. 为什么在面向对象设计中需要三种对象？
3. 什么是导航能力？请列举一个导航能力的例子。
4. 什么是面向对象设计中的可见性？解释不同层次的可见性。
5. 对象复用的关键原因是什么？
6. 开发人员使用哪些方法实现对象复用？
7. 面向对象设计有哪些主要活动？
8. 精炼对象设计中的用例模型的目标是什么？为什么重要？
9. 我们可以使用什么方法确定用例设计对象，即接口对象、控制对象和实体对象？
10. 构造对象责任图的目标是什么？此图由什么部分构成？
11. 在确定对象行为和对象的责任中我们应该寻找什么？

12. 对象状态和状态转换事件之间是什么关系?
13. 构造状态图需要哪些步骤?

问题和练习

1. 使用面向对象方法开发系统的主要原因是什么? 为什么?
2. 用 PL/1 开发一个项目预期要花 30 个月。假定与

14. 使用哪些工具记录用例详细的对象交互?
15. 一个设计类图包含什么?

下表显示的比率相同, 比较用 PL/1 开发项目和用类似 Smalltalk 的面向对象语言开发项目的开发周期、工作量和软件大小。

编程语言	项目周期 (自然月)	工作量 (人月)	软件大小 (代码行)
PL/1	30.0	240.0	41 800
面向对象语言	5.5	16.4	3 500

3. 判断题。如果需要请解释你的回答。
 - 依赖关系只在两个实例之间建模了两个类关联关系。
 - 为了强化封装, 属性一般应该定义为私有的。
 - 假设一个对象同其他对象协作, 当需要的时候提供请求服务, 但它无法实现这些, 则称为不负责任的对象。
 - 接口对象典型情况下是持续性的。
 - 在面向对象设计阶段, 修改对象模型以反映实际的实现环境。
4. 对于以下内容, 用户可以发现哪些接口对象?
 - a. 不需要计算机来打印照片的照片打印机。
 - b. 服务站加油泵。
 - c. 零售店的出入口。
5. 在面向对象设计的设计阶段, 对早先创建的用例要进行哪些修改? 如果需要修改, 修改什么, 修改的目的是什么?
6. 选择一个你熟悉的应用。挑选应用中的一个过程, 并创建一个分析用例; 使用图 17-7 的模板。

然后, 使用本章的设计指南, 精炼该用例并把它转换成设计用例。加粗你修改或增加的区域。

7. 创建设计用例之后, 分析它以确定和分类用例设计对象, 以图 17-8 为例。一般来说, 获得的实体对象要比接口对象多, 而且至少应该有一个控制对象。让一个同学或同事检查你的工作, 以确保对象被正确地确定和分类。
8. 对象责任图的目的是什么? 图中使用什么符号, 它们分别代表什么? 下一步, 基于你的用例绘制一个对象责任图, 以图 17-9 为例。
9. 现在, 回到你创建的设计用例。分析这个用例以确定所需的系统行为, 以图 17-10 显示的矩阵为例。确定了用例行为后, 决定每个行为是要在新系统中自动完成还是手工完成。如果行为需要自动完成, 则在第三列中指定负责执行该行为的对象类型。
10. 解释类责任协作 (CRC) 卡的目的; 然后为你在前面练习中确定的每个对象类创建一个 CRC 卡。

项目和研究

1. 使用你在问题和练习中创建和细化的设计用例 (或者另一个)。基于这个用例创建一个顺序图和类图; 以图 17-16 和图 17-17 为例。让你所在企业或学校的一些熟悉面向对象设计的人检查你的图, 如果需要请修改。
2. 设想实现一个假想系统。创建描述系统软件和硬件的物理架构的组件图和部署图, 以图 17-18 和图 17-19 为例。让你所在企业或学校的一些熟悉使用面向对象技术的人检查你的图, 如果需要请修改。
3. 正如面向对象分析 (OOA) 导出并转换到面向对

象设计 (OOD), OOD 也将导出并转换到面向对象程序设计。尽管这不是一门程序设计课程, 但理解面向对象程序设计 (OOP) 的基本概念和结构可能会对系统开发的面向对象方法的整体理解有益。研究网上或者课本中的面向对象程序设计内容。在面向对象分析和设计阶段创建的图形、用例和产品如何在构造阶段被面向对象程序设计使用? 面向对象程序设计引入了什么新的图形和结构吗? 在构造阶段, OOP 使用什么基本步骤和过程? 如今使用哪些最流行的面向对象程序设计语言?

小型案例

1. 获得 Jim Conallen 的书《Building Web Applications with UML》(Boston: Addison-Wesley, 2002):
 - a. 在 Conallen 的书中基于 Web 的 UML 和传统的 UML 有什么差别?
 - b. Wow Munchies 公司 (在前一章中讨论的) 已经决定开发一个电子商务站点。你愿意为这个网站进行 UML 建模, 你将使用什么 UML 建模技术? 为什么?
2. 在前一章中, 你同一个政府部门交谈, 并开始为它设计一个新系统。
 - a. 描述你推荐的系统。
 - b. 使用 UML 建模, 绘制你建议的系统 (考虑用例图、类图、顺序图 and 状态机图)。
 - c. 给你的老师提交一个成果, 包括前面交谈的讨论和本章小型案例的第 2a 部分和第 2b 部分。你将被按照正确性、完整性和专业性打分。
3. 使用小型案例 2 和前面章节中的工作, 为你的政府客户创建一个系统原型。你将使用什么语言? 为什么? 用 CD 提交你的原型给老师, 以及你的源代码的拷贝、屏幕截屏和你解决的业务问题的简单讨论。

团队和个人练习

1. 圆桌讨论: 现在你就要完成本课程了, 总结系统分析和设计的领域, 以及你的经验。考虑以下内容: 什么是系统分析和设计? 对于一个这个领域的人来说具有什么品质是最重要的? 你从本课程中学到了什么? 如果你有机会改变一些事情, 你会怎么做?
2. 在整个课程中, 你得到鼓励通过一些练习增强你的创造力。创造力、自由思考和反常规的能力对于许多领域的实际成功很重要。为什么?
3. 团队或个人练习: 考虑规则错误、低效或者过时的情况。在课堂上研究并共享挑战系统和改变规则的法律方法。可以是政府的 (法律)、工作场所的或者学校的。

系统分析和设计完成后的工作

本部分介绍系统开发的最后阶段。

第 18 章（系统构造和实现）介绍从物理设计说明构造系统以及构造后的系统投入实现的过程。

系统构造和实现

本章概述和学习目标

在本章中，你将学到许多有关系统开发的构造和实现阶段的内容，这两个阶段构造、测试、安装并交付最终的系统投入运行。本章将具体介绍如下内容：

- 解释系统生命周期的构造和实现阶段的目的。
- 按照主要任务、角色、输入和输出描述系统构造和实现阶段。
- 解释几种应用程序和系统测试。
- 确定几种系统转换策略。
- 确定本书中有助于实际执行系统构造和实现任务的章节。

尽管本章介绍了一些系统构造和实现的技术，但是讲授这些技术并不是本章的目的，本章只介绍系统构造和实现的过程。

本章关键术语

系统构造（system construction）是系统组件的开发、安装和测试。

模块测试（stub test）是对程序的一个子集进行的测试。

单元或程序测试（unit or program test）是对整个程序的测试。

系统测试（system test）是对整个系统的测试。

系统验收测试（system acceptance test）是对最终系统进行的测试，其中用户进行验证测试、确认测试和审计测试。

审计测试（audit test）是确保系统准备就绪可以运行的测试。

18.1 什么是系统构造和实现

下面开始介绍系统构造和实现。**系统构造**是系统组件的开发、安装和测试。然而，系统开发是常见的同义词（我们不喜欢这个同义词，因为它更常用于描述整个生命周期）。**系统实现**是交付系统投入运行（意思是日常运行）。

图 18-1 说明了系统构造和实现阶段，注意系统构造阶段的触发器是设计阶段得到的对物理设计说明复用的认可。给定设计说明后，我们就可以构造并测试设计中的系统组件了。最后，我们将构建实际的系统，然后这个实际系统可以作为一个运行系统实现和交付。

本章详细介绍各个阶段。

18.2 构造阶段

构造阶段的目的是开发和测试一个实现了业务需求和设计需求的功能系统，并实现新系统和现有生产系统的接口。一般认为编程是构造阶段的主要内容，但是越来越多的系统采用采购和购买软件包方案，软件组件的实现和集成正变成构造阶段中同等重要（如果不是更重要的话）的内容。

在本节中，你将了解一个典型的系统开发项目构造阶段中涉及的几个任务。图 18-2 是一个描述了构造阶段的各项任务的活动图。下面详细地介绍构造阶段的每个任务。

18.2.1 任务 6.1——构建和测试网络（如果需要）

在系统分析的需求分析阶段，确定了网络需求；随后，在设计阶段，开发了分布式数据和过程模型。使用这些技术性设计说明实现一个新信息系统的网络架构是构造和实现活动的前置条件。

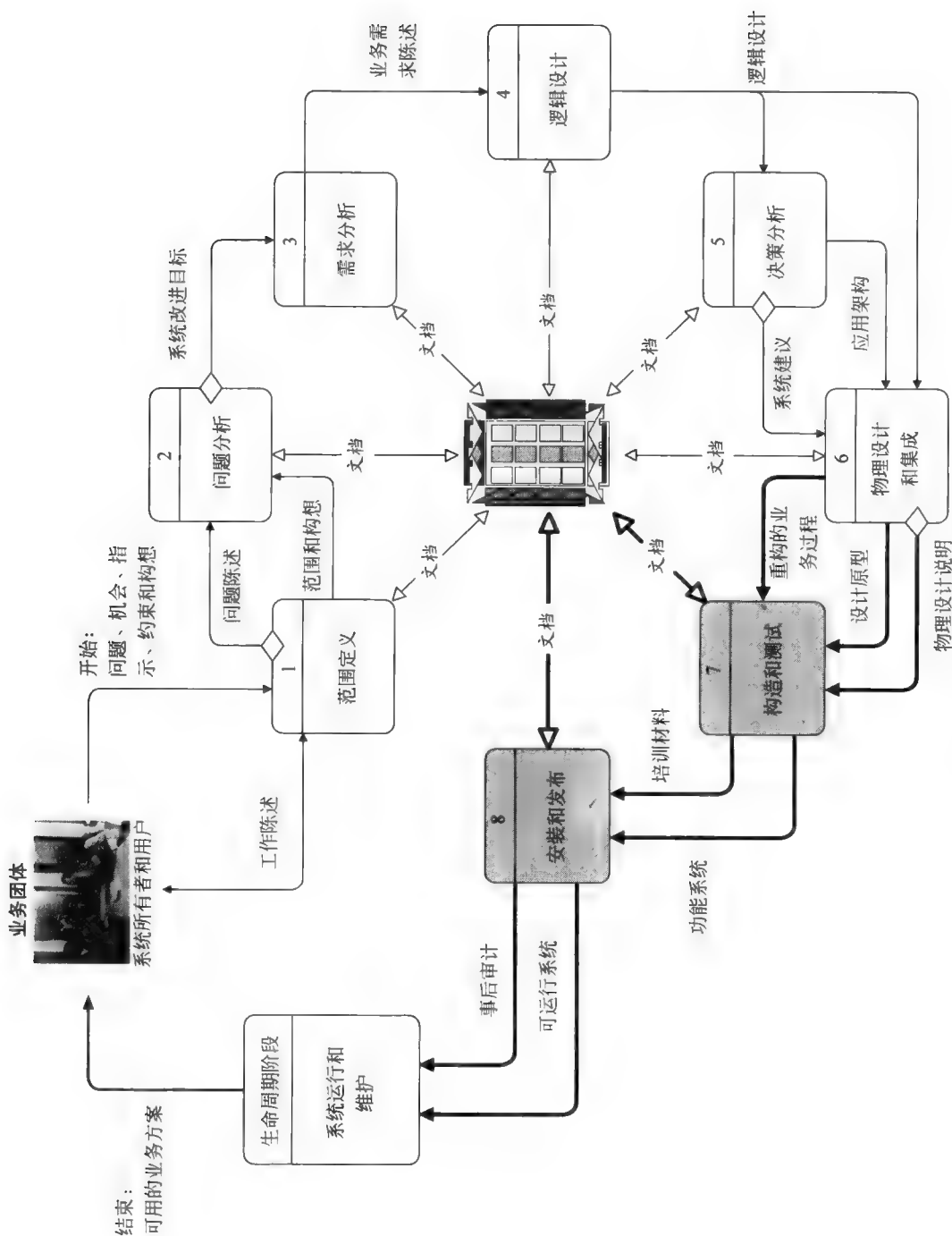


图 18-1 系统构造和实现的上下文

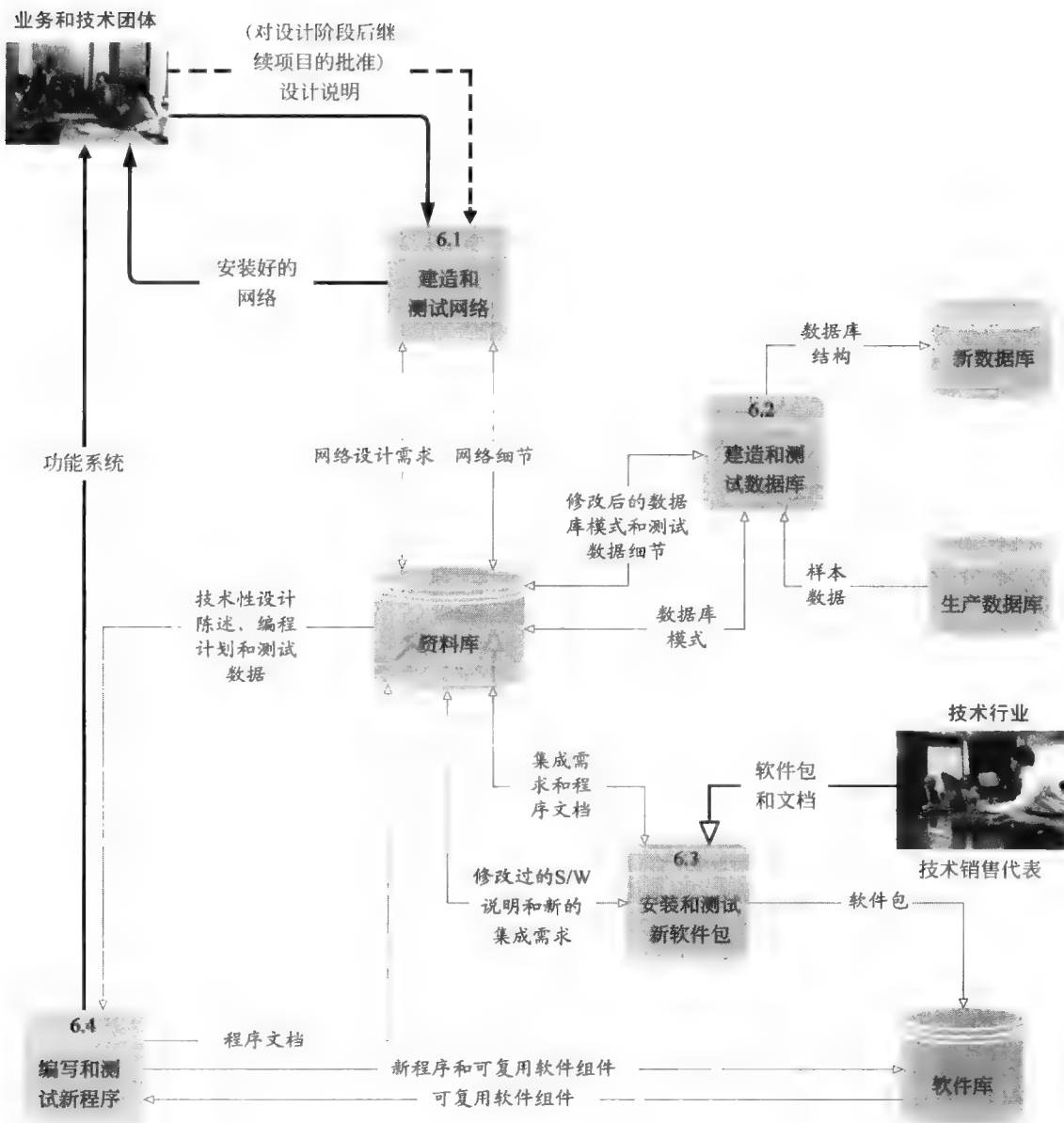


图 18-2 系统构造任务

在许多情况下，新的或者增强的应用系统围绕现有网络建造。如果是这样，就可以忽略这个任务。但是，如果新应用要求有新的网络，或者修改现有网络，通常网络就必须在构建和测试数据库以及编写和安装将使用那些网络的计算机程序之前被实现。因此，构造阶段的第一个任务可能是构建和测试网络。

这个阶段涉及分析员、设计人员和构造人员，网络设计人员和网络管理员主要负责完成这个任务，网络设计人员是设计局域网和广域网及其连接的专家。网络管理员具有构建和测试用于新系统的网络技术的专业知识，他（或她）也应该熟悉任何可能的新的联网技术必须遵循的网络架构标准，这个人还可以负责网络安全（网络设计人员和网络管理员可以是同一个人）。虽然系统分析员可以参与这个任务，但是分析员的角色更多的是推动者——确保网络方案满足业务需求。

18.2.2 任务6.2——构建和测试数据库

构建和测试数据库是许多学生不熟悉的任务，他们习惯了由老师给他们提供测试数据库。这个任务必须在其他编程活动之前进行，因为数据库是要编写的计算机程序的共享资源。如果新系统要求新的或者修改的数据库，则现在就可以构建和测试那些数据库。

这个任务涉及系统用户、分析员、设计人员和构造人员，设计数据库的同一系统专家将主要负责完成这个任务，系统用户也可以参与，他们提供或者认可用在数据库中的测试数据。当要构建的数据库是面向应用的独立数据库时，则经常由系统分析员完成这个任务；否则，系统分析员的任务最多是确保系统符合业务需求。数据库设计人员将经常成为负责完成这个任务的系统构造人员，这个任务可以请数据库程序员构建初始的数据库，还可以请数据库管理员调试数据库性能、增加安全控制并提供备份与恢复。

这个任务的主要输入是系统设计阶段制定的数据库模式。来自生产数据库的样本数据可以被加载到表中，供测试数据库之用。这个任务的最后产品是新数据库的一个未填充数据的数据库结构。未填充数据这个词意味着实现了数据库结构，但数据没有被加载到数据库结构中。如下面介绍的，程序员最终会编写程序来加载数据，并维护新数据库。修改后的数据库模式和测试数据细节也在这个任务期间产生，并放置在项目资料库中供日后参考。

18.2.3 任务6.3——安装和测试新软件包（如果需要）

有些系统方案可能要求购买或者租用软件包。如果是这样，一旦新系统的网络和数据库建成，我们就能够安装和测试新软件，这个新软件以后将被放置在软件库中。

这个活动一般涉及系统分析员、设计人员、构造人员以及供应商和咨询顾问，这是生命周期中的第一个应用编程任务。通常系统分析员参与软件包的测试，他们的任务是澄清需求。同样，系统设计人员也可以参与这个任务，他们澄清用在测试软件中的集成需求和程序文档。网络管理员可以参与在网络服务器上实际地安装软件包。最后，这个任务一般都需要软件供应商和咨询顾问参与，他们辅助安装和测试过程。

这个任务的主要输入是从系统供应商处收到的新软件包和文档，应用程序员将按照系统设计期间开发的集成需求和程序文档完成软件的安装和测试。这个任务的主要交付成果是安装和测试后的软件包，可以从软件库中得到它们。任何修改后的软件说明和所需的新集成需求都归档到项目资料库中，作为一个历史记录供日后参考。

18.2.4 任务6.4——编写和测试新程序

我们现在准备好了为新系统开发（或者完成）内部程序。原型程序经常在设计阶段构造，可以把这些原型包括进来作为完成系统构造和实现的一部分技术性设计说明，但是这些原型一般功能不全（或者完整），所以这个活动可能涉及开发或者精炼那些程序。

这个任务涉及系统分析员、设计人员、构造人员。系统分析员一般澄清程序要实现的业务需求，系统设计人员可能要澄清用在编写和测试程序中的程序设计、集成需求和程序文档（系统设计期间开发）。系统构造人员将对这个活动负主要责任，应用程序员（构造人员）负责编写和测试内部软件。绝大多数大型编程项目需要团队工作。一种流行的组织策略是使用主程序员团队：团队由经验丰富的主程序员管理，主程序员负责整个程序设计策略、标准和构造，检查所有的编码和测试活动，并解决程序最困难的方面；其他团队成员包括备份主程序员、程序资料员、程序员和专家。应用程序员经常通过使用软件测试器提供帮助，软件测试器专门用于构建并运行测试脚本，测试脚本一致地应用于程序上以测试所有可能的事件和响应。

这个活动的主要输入是技术性设计陈述、编程计划和系统设计期间开发的测试数据。由于其他现有系统可能已经编写完成，并使用了某些新程序或者程序组件，所以有经验的应用程序员将首先检查软件库中可以获得的可复用软件组件。这个活动的主要交付成果是新的程序和放置在软件库中的可复用软件组件，也得到一个需要由质量保证小组认可的程序文档。一些信息系统开发商拥有由专家构成

的质量保证小组，他们评审最终的程序文档对标准的符合性，并提供有关质量建议和需求的相应反馈。最终的程序文档放置在项目资料库中供日后参考。

测试是一个经常在有关计算机编程的专业课程中被忽略的重要技能，它不应该推迟到整个系统已经编写完了才进行。通常存在3个级别的测试：模块（stub）测试、单元或程序测试及系统测试。**模块测试**是测试程序的单个事件或模块。换句话说，它是对程序的孤立子集的测试。**单元或程序测试**是在程序的所有事件和模块被编码并通过了模块测试后，所进行的整体单元测试，也就是测试整个程序。**系统测试**用于确保编写和独立测试的应用程序集成到整个系统中时能够正确工作。为测试系统应该开发一个系统测试计划，并遵循之。为每个功能和非功能需求开发一个或多个测试脚本。

单个程序工作正常并不意味着它能够与其他程序一起工作正常。集成的程序集应该通过系统测试来确保程序正确地接收（作为输入）另一个程序的输出。一旦系统完成并认为成功，下面就可以继续进行系统实现。

18.3 实现阶段

剩下要做什么？新系统通常会表现出对企业当前工作方式的某种不适应，所以，分析员必须提供从旧系统到新系统的平稳转换，并帮助用户应付常见的启动问题。因此，实现阶段交付系统运行。

来自构造阶段的功能系统是实现阶段的关键输入（见图18-1）。实现阶段（和项目）的交付成果是将要进入生命周期的运行和支持阶段的运行系统。

在信息系统框架中，实现阶段考虑与构造阶段相同的构件。在本节中，你将了解到一个典型系统开发项目的实现阶段中涉及的几个任务。图18-3是一个描述了实现阶段的各项任务的活动图。下面将详细地介绍每个实现阶段任务。

18.3.1 任务7.1——进行系统测试

现在，软件包和内部程序已经被安装和测试，我们需要进行最后一次系统测试。所有软件包、定制程序和任何包含新系统的现有程序都必须被测试，以确保它们能够一起工作。

这个任务涉及分析员、所有者、用户和构造人员。系统分析员推动任务的完成，他们常常同项目团队成员就测试问题进行沟通。系统所有者和系统用户对系统是否正确工作拥有最终决定权。系统构造人员和各种专家参与系统测试，例如：可能需要应用程序员、数据库程序员和网络专家解决系统测试期间暴露的问题。

这个任务的主要输入包括软件包、定制程序以及任何包含新系统的现有程序。系统测试使用前面由系统分析员开发的系统测试数据进行。如同前面进行的测试，系统测试可能导致要求对程序进行修改，于是，再一次回到某个构造阶段任务。这个迭代将继续下去，直到完成一次成功的系统测试为止。

18.3.2 任务7.2——准备转换计划

一旦完成了一次成功的系统测试，我们就可以开始准备让新系统投入运行。系统分析员将使用新系统的设计说明开发一个详细转换计划，这个计划确定要安装的数据库、需要开发的用户培训和文档以及从旧系统到新系统的转换策略。

项目经理推动这个活动。系统分析员、系统设计人员和系统构造人员一般不参与其中，除非项目经理要求他们参加。最后，许多组织要求所有项目计划正式地向一个指导部门（有时称为指导委员会）汇报，以获得最后批准。

这个活动由一次成功的系统测试的完成触发。可以使用新系统的设计说明制定一个详细转换计划。这个活动的主要交付成果是转换计划，它将确定要安装的数据库、需要开发的用户培训和文档以及从旧系统到新系统的转换策略。

转换计划可以采用以下常用的安装策略之一：

- **突然切入。**在一个特殊的日子（通常和一个办公业务周期重合的日期，例如月、季度或者财政年度），旧系统终止，新系统投入运行。这是高风险的方法，因为仍然会存在严重的问题，直到系统运行了一段时间后才会发现。但另一方面，这种方法没有转换费用。例如，如果某个政府

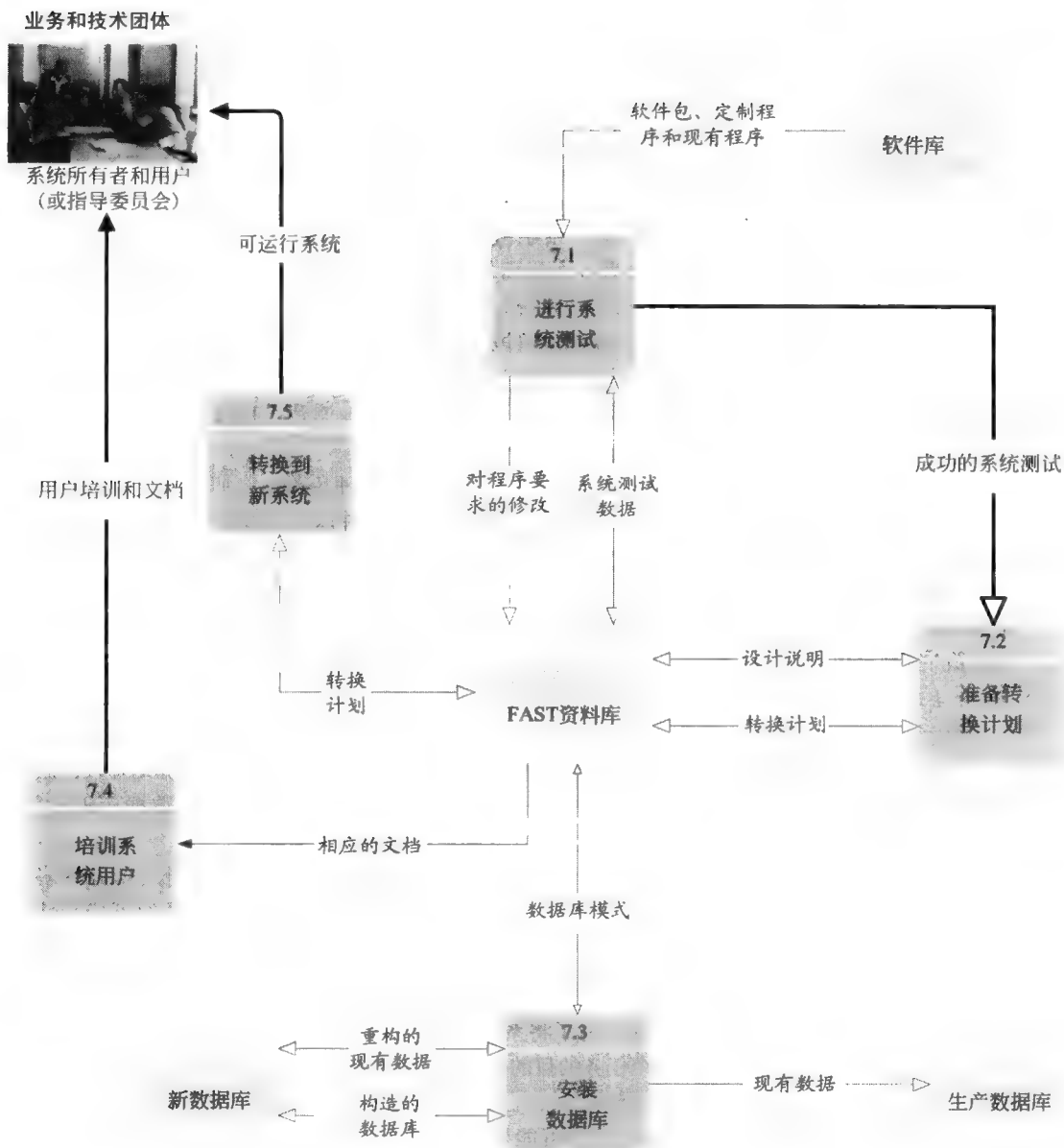


图 18-3 系统实现的任务

强制性规定或者业务策略在某个特定的日子开始生效，而系统不能在那一天以前实现，就可能不得不采用突然切入。

- **并行转换。**在这种方法下，旧系统和新系统同时运行一段时间，这确保了新系统中所有严重的问题在旧系统被丢弃前都被解决了。最后的切入可以是突然的（通常在一个业务周期之后），也可以是渐变的（随着新系统各个部分的完善逐渐切换）。这个策略最小化了新系统中的大错误对业务造成不可恢复伤害的风险，但是，它也意味着必然导致一段时间运行两个系统的费用。因为在计算机上运行同一个系统的两个版本可能会对计算资源提出不合理要求，所以这种方式只有在旧系统主要是手工系统时才可行。
- **位置转换。**当相同的系统将用于多个地理位置时，通常首先在一个位置上转换（使用突然切入或者并行转换）。一旦该地点认可了系统，系统就可以应用到其他地点，而其他地点可以采用

突然切入方式，因为大的错误已经被修改。进一步来说，其他地点可以从学习第一个测试地点的经验中获益。第一个测试地点经常称为 β 测试地点。

- **分阶段转换。**类似于位置转换，分阶段转换是在突然切入和并行转换上的变化，它基于前面介绍的版本概念。新系统的每个后续版本在开发后就进行转换，每个版本都可以使用突然切入、并行转换或者位置转换。

转换计划通常还包括一个系统验收测试计划。系统验收测试给最终用户、管理人员和信息系统操作管理员最后一次机会决定接受或者拒绝系统。**系统验收测试**是最终用户使用真实数据一段时间后进行的最终系统测试，这是一种详细测试，涉及3个层面的验收测试：验证测试、确认测试和审计测试。

- **验证测试**在模拟环境下使用模拟数据运行系统，这个模拟测试有时候称为 α 测试，它主要查找错误和遗漏，这些错误和遗漏往往与在前面阶段说明了但没有在构造期间实现的最终用户和设计说明有关。
- **确认测试**在实际环境中使用真实数据运行系统，这有时称为 β 测试。在这个确认过程中，可以测试以下一些项目：
 - a. **系统性能：**系统处理的吞吐量和响应时间足够满足正常处理负载吗？如果不能，可能需要重写一些程序来提高效率，或者可能需要替换或升级处理硬件以处理额外负载。
 - b. **峰值负载处理性能：**系统可以处理峰值期间的负载吗？如果不能，可能需要提高硬件和/或软件的效率，或者可能需要重新调度处理过程，即考虑在非峰值期间进行一些不太重要的处理。
 - c. **人类工程学测试：**系统如预期的那样易学易用吗？如果不是，是否基本够用呢？对人类工程学的改进可以推迟到系统投入运行之后进行吗？
 - d. **方法和程序测试：**在转换过程中，将对新系统的方法和程序进行第一次的真正测试。如果用户觉得它们笨拙而低效的话，可能必须修改方法和程序。
 - e. **备份和恢复测试：**应该测试所有的备份和恢复程序，这应该包括模拟数据损失灾难，然后测试从灾难中恢复所需的时间。而且，也应该进行数据的前后比较，以确保数据正确地恢复。测试这些程序很重要，不要等到灾难发生才发现恢复程序中有错误。
- **审计测试**证实系统没有错误并准备就绪可以运行。不是所有的组织都要求审计，但是许多公司拥有独立的审计或质量保证人员，他们必须在系统最后投入运行之前证实系统的可接受性和文档。有一些独立的公司为用户组织进行系统和软件证实。

18.3.3 任务7.3——安装数据库

在前面的阶段中，构建并测试了数据库。为了将系统投入运行，还需要加载数据库。所以，我们将要介绍的下一个任务就是数据库安装，这个任务的目的是在新系统数据库中添加旧系统中的现有数据。

最初，这个活动可能看上去是微不足道的，但是请考虑加载一个典型的表的情况，例如：MEMBER表。在数据库表准备好投入运行之前，可能要加载几十个、上百个或者上千个记录，而且每个记录都必须被输入、编辑和验证。

系统构造人员在这个活动中扮演主要角色。这个活动一般由应用程序员完成，他们编写特殊的程序从现有数据库中提取数据并载入（填充）到新数据库中。系统分析员和设计人员在这个任务中可能只扮演一个小角色，他们将主要参与计算数据库大小和估计安装所需的时间。最后，一般分配数据录入人员或者雇用的帮手来做数据录入工作。

为了填充新数据库，需要编写特殊的程序，也就是需要编写计算机程序重构来自生产系统的现有数据，并用重构后的数据填充新数据库（生产系统的数据库与新数据库的数据库模式模型和数据库结构耦合）。这个任务的主要交付成果是重构后的现有数据，将它们填充到新系统的数据库中。

18.3.4 任务7.4——培训用户

变革可能是件好事，但它并不总是那么容易。转换到一个新系统需要培训系统用户，并提供指导用户如何使用新系统的文档（用户手册）。

培训可以一个接一个地进行，但是通常首选小组培训。小组培训可以更好地利用时间，并且有利于

小组学习。考虑一下你受过的教育就明白了，你从同学和同事那里学到的实际上比从老师那里学到的要多。老师推动了学习，但你通过与一大群人实践来掌握特定技能，在那里可以更加有效地涉及公共问题。小组培训可以产生教育的“水波效应”——第一组接受培训的人员以后可以培训其他几组人员。

这个任务由系统分析员完成，并且涉及系统所有者和用户。根据新系统的相应文档，系统分析员将提供用户文档（一般以手册的形式）和对系统用户的培训。系统所有者必须支持这个活动。他们必须愿意让人们有时间获得成为新系统的成功用户所需的培训。记住，系统是为用户设计的。用户的参与在这个活动中很重要，因为用户将从这个活动中学到成功与失败的经验教训。好在人们很少忽视这个任务期间的用户参与。最重要的用户参与是培训用户并向他们提出忠告——他们必须接受培训，从而能够正确地使用设备和遵守新系统要求的工作程序。但是无论培训得有多好，用户有时还是会感到不清楚，或者也许他们会发现错误或限制。所以，分析员有责任帮助用户通过学习阶段直到他们熟悉新系统为止。

根据新系统的相应文档，系统分析员将给系统用户提供正确地使用新系统的文档和培训。这个任务的主要交付成果是用户培训和文档。许多组织雇用特殊的系统分析员，他们只编写用户文档和培训指南。如果你有能力清晰地写作，你的工作机会就在那里！图 18-4 是一个典型培训手册的提纲。编写用户手册的原则是：“站在别人的角度写作，就好像让他们为你写作一样。”你不是一个业务专家，更不要期望读者是一个技术专家。每种可能情况及其正确的操作程序都必须记录下来。

培训手册最终用户指南提纲	
I. 引言	2. 第一次使用
II. 手册	a. 开始
A. 手册系统（详细地解释人们的工作和新系统的标准操作程序）	b. 课程
B. 计算机系统（它如何用于整个工作流程）	C. 参考手册（对于非初学者）
1. 熟悉终端/键盘	III. 附录
	A. 错误提示消息

图 18-4 一个培训手册的提纲

18.3.5 任务 7.5——转换到新系统

从旧系统转换到新系统是一个重要的里程碑。转换之后，系统的所有权正式从分析员和程序员转移到用户。分析员通过仔细地执行转换计划完成这个任务，转换计划包括详细的安装策略，从现有系统转换到新的生产信息系统时必须遵循此计划。这个任务还包括全部的系统审计。

这个任务涉及系统所有者、用户、分析员、设计人员和构造人员，监督这个转换过程的项目经理主持该任务。系统所有者提供对整个项目的反馈意见，也可以提供对投入运行的新系统的反馈意见。系统用户将提供对新系统的实际使用有价值的反馈意见，这些反馈意见是用来度量系统的接受程度的主要依据。一旦系统投入运行，系统分析员、设计人员和构造人员就将处理从系统所有者和用户那里收到的各种反馈意见。在许多情况下，这些反馈意见可能会触发改正缺陷的行动。无论如何，反馈意见将一直用来帮助评测新的系统项目。

这个活动的关键输入是在前面的实现阶段任务中创建的转换计划，主要交付成果是投入企业生产的运行系统。

复习题

- 构造阶段的主要目的和主要活动是什么？
- 谁是网络设计人员和网络管理员？
- 当构建和测试数据库时，需要完成什么任务？
- 谁参与新软件包的安装和测试？他们的工作是什么？
- 什么是首席程序员团队？
- 本书推荐哪三种测试？
- 为什么需要实现阶段？
- 在实现阶段，一般谁参与系统测试？
- 4 种常用的转换策略是什么？
- 使用直接切换作为转换策略有哪些潜在的问题？
- 使用并行转换作为转换策略有哪些潜在的问题？

- 12. α 测试和 β 测试有什么区别？
- 13. 在安装数据库时谁是主要的参与者？他们的责任是什么？

问题和练习

- 1. 你是一个大型企业系统的系统测试团队的首席分析员，该系统几乎涉及企业的每个业务功能。不幸的是，设计和构造推迟了进度大约两周时间。假设在没有严重问题的情况下，系统测试计划要花 4 周。增加资源不会缩短所需的时间。如果你按计划执行，实现将延迟两周。系统所有者（也是 CEO）觉得这是不可接受的，并告诉你：“将要花一个月进行系统测试，你的意思是什么？我需要这个系统在两个星期内运行起来，一天也不能推迟。如果你发现有什么问题，可以在以后解决！”你在这种情况下应该怎么做？
- 2. 考虑前一题的另一种情况。你作为开发这个企业系统的软件开发商的测试分析员。如果项目没有按进度完成，你的公司将失去额外的奖金。由于设计和构造落后于进度，你将不得不将系统测试减少一半。你的公司对你施加了强大的压力要你压缩测试，以便项目可以按进度完成，公司将收到奖金。你已经争辩说如果压缩测试，即使采取基于风险的测试策略，也可能会遗漏某些严重的问题。你将如何做？
- 3. 你是一个将要领导另一个项目的系统测试团队的系统分析员。你的公司采用了一种新的测试策略。过去，构造系统的程序员自己进行系统测试。为什么这不是一个好主意？
- 4. 你应该为你的系统测试团队选择什么人员？他们应该拥有什么技能？
- 5. 以下陈述是对是错？如果需要，解释你的回答。
 - a. 构建和测试所需的数据库应该在编程活动结束后。
 - b. 用户培训应该在构造之前很多时候进行，以便确保每个人都获得培训而没有遗漏。
 - c. 并行转换的目的是降低业务风险。
 - d. 测试是高度结构化的活动，除非整个应用编写完成，不应该急于进行。
 - e. 系统开发和系统构造经常用作同义词，但它们可能不是指同一个意思。
- 6. 作为一位系统分析员，你参与了为你的企业服务部门开发一个库存跟踪系统的项目。现在项目进行到了最后阶段，要求你写一份培训手册。使用图 18-4 显示的提纲，编写部分的培训手册（一

- 14. 当培训用户时系统分析员有什么责任？
- 15. 即使新系统完全实现了并且功能正常，为什么还需要反馈？

- 至两页）描述这个手工系统或计算机系统。让你的一个同学或者同事阅读你写的部分，并评价其清晰性。他觉得你写的能理解并且清楚吗？提供了最终用户需要的合适程度的细节吗？
- 7. 在系统构造和实现期间，并不是所有的活动都是技术上很自然而然的，所以除了系统测试以外用户不需要参与吗？
 - 8. 匹配第 1 列中的词汇和第 2 列中的定义或例子：

1. Beta 测试	A. 没有加载数据的产品数据库
2. Alpha 测试	B. 在正常负载下的吞吐量/响应时间测试
3. 程序测试	C. 移植整个系统进入生产环境
4. 审计测试	D. 未预料到的系统突然死机测试
5. 系统性能测试	E. 应用程序级的代码测试
6. 未装载的数据库	F. 独立进行的认证等级测试
7. 备份和恢复测试	G. 模块层次的代码测试
8. 峰值性能测试	H. 详细的验证、认证和审计测试
9. 直接切换	I. 应用程序环境层面的测试
10. 系统实现	J. 用户使用模拟数据进行的环境层面的测试
11. 系统验收测试	K. 用户使用真实数据进行的现场环境层面的测试
12. 系统测试	L. 在峰值负载下的吞吐量/响应时间测试
13. 模块测试	M. 安装策略类型

- 9. “人机界面设计的目标是创建一个易用的系统。需要使用用户手册就是设计上的失败。”回应这句话。你同意其观点吗？为什么？
- 10. 许多企业要求一份实现后评估报告（PIER），通常在实现后 6 个月到 1 年时间里。这份报告的目的是什么？
- 11. 如果一个项目设计和构造得很差，规划良好并且执行良好的实现工作会对项目的成功有所帮助吗？相反的情况会怎么样？设计良好和构造良好的系统会克服差的实现工作吗？

项目和研究

- 一些公司, 例如 Mercury Interactive of Rational (现归 IBM 所有), 提供自动化的软件测试包作为独立产品, 或者作为更大的软件包的一部分。在网上和商业期刊上研究这些软件包。下载并试用你找到的试用版本。另外, 联系几个本地企业中的软件测试人员, 同他们讨论他们的软件测试方法。
 - 描述你的研究——你找到了什么产品?
 - 比较它们的特点和功能。
 - 你联系的软件测试人员使用自动化的软件测试工具吗? 如果使用, 是自己研制的工具还是商业产品? 他们表示出认为哪一个工具是最好的吗?
 - 如果你是测试经理, 并且有权选择购买自动化的软件测试工具, 你会选择哪一个? 还是你会倾向于开发自己的自动化软件测试工具? 解释你的回答。
 - 你认为使用自动化软件测试包的主要优点是什么?
- 你是为一个企业开发的大型项目中工作的系统分析员, 该企业在总部有几百个雇员, 大约 12 个办事处分布在国内、加拿大和墨西哥。项目的目的是实现一个企业级的关键任务信息系统。项目目前正处于构造和实现阶段, 你被分配负责选择转换策略并编写转换计划。准备一份本书中讨论的不同的安装策略总结分析报告, 并推荐一个你认为最合适的策略。
- 假设你的推荐被管理层采纳, 起草一份详细的转换计划, 包括转换到新系统的实际的转换策略。完成计划起草后, 你让几个在开发转换策略方面有经验的 IT 职员检查它。进行所需的修改, 然后再次检查, 直到大家都认为你的计划现实可行为止。
- 下一步是准备系统接受性测试计划。使用本书中的材料作为指南, 在网上调研一些更详细的进行接受性测试的内容。选择一些可得到的测试模板, 根据需要修改它们。然后起草测试计划, 并同 IT 测试人员讨论。确保你的计划涉及到了任何潜在的风险。根据需要进行修改并再次检查, 直到你的计划准备好可以实施。
- 本书描述了一种传统的进行最终用户培训的方法。还有其他的方法 (例如基于 Web 的) 可以提供更有效的用户培训吗? 调研一些正在变得越来越广泛使用的基于 Web 的培训方法。然后使用问题 2 描述的场景, 开发一个最终用户培训计划。完成计划起草后, 你让一些专业的培训人员检查其完整性和可行性, 并根据需要做出修改。然后让你的一些同学或者同事从一个用户的观点检查你的培训计划。你制定了一些可行的计划了吗?
- 有一个不成文的原理: 无论你对系统实现做了多少计划, 总会有一些未预料到的事情发生, 而且经常是在最糟糕的时候发生。同几个本地企业中在实现系统方面有经验的分析员交谈。询问他们的经历, 他们经历过的最糟的情况以及他们从中学到了什么。然后使用你掌握的信息组织一套关于在系统实现期间及计划和处理未预料事件的指南。

小型案例

- Wow Munchies 的 Web 站点是 www.wowmunchies.com, 目前放在一个称为 Cool Hosting 的 Web 主机服务公司的 123coolhost 服务器上。但 Wow Munchies 决定让另一个主机服务公司: Reliable Host, 使用 123reliable 服务器更新 Web 页面和提供服务。新的主机服务公司在加载并测试了 Web 页面之前, 将 www.wowmunchies.com 的 DNS 指

向了 123reliable 服务器。DNS 变更花了 12 ~ 72 个小时, Reliable 公司想在这个期间将能够让 Web 页面启动起来, 但没有启动起来。结果, 在新的站点再次工作之前的几天, DNS 已经指向了新的服务器。由于网站宕机, Wow Munchies 损失了大约 200 000 美元。评价哪个地方出错了, 应该如何避免。

团队和个人练习

- 个人练习: 在课堂上展示一个发布物 (任何发布物)。穿着职业化, 使用有趣的展示技术。记住要穿着职业化、语速慢而清晰, 并具有幽默感。没有人比你更了解你的发布物!
- 教师/班级练习: 天气允许, 课外活动。
- 个人/班级练习: 创建一个班级中每个人 (以及其他任何想加入的学生) 联系信息的网络化表单。提交信息的每个人都获得一份拷贝。离开学校后保持联系!

CASE 资料库 (CASE repository) 是一个系统开发人员的数据库。它是开发人员存储系统模型、详细描述和说明以及系统开发的其他产品的地方。资料库的同义词包括字典和百科全书。

PERT 图 (PERT chart) 是一种图形化的网络模型，描述一个项目中任务之间的关系。

Web 服务器 (web server) 是运行因特网或内联网站点的服务器。

版本控制 (version control) 是指对程序修改的跟踪。

办公自动化系统 (office automation system) 支持广泛的企业办公活动，改进工作人员之间的工作流的信息系统。

报价申报书 (request for quotation, RFQ) 是一种与单个软件供应商交流应用软件包的业务、技术和支持需求的正式文档，该软件供应商已经被选中提供应用软件包或服务。

变动成本 (variable cost) 是与某些使用因素成比例的费用。

变化管理 (change management) 策略建立一个过程以管理在项目期间出现的变化。

表 (table) 是文件在关系数据库中的等价。

表查询文件 (table look-up file) 是包含相对静态的可被共享的数据的表。

表输出 (tabular output) 是以文本和数字列的形式表现的输出。

菜单驱动 (menu driven) 是一种对话策略，要求用户从选项菜单中选择动作。

参与者 (actor) 代表了需要同系统交互以交换信息的任何事物。

参与者 (actor) 是任何需要同系统交互的事物。

策略 (policy) 是一套约束业务过程的规则。

差距分析 (gap analysis) 是将对商用软件包的业务和技术需求与特定商用软件包的功能和特征进行比较，以定义不能满足的需求。

超类 (supertype) 是一个实体，其实例存储了一个或多个实体子类的公共属性也称抽象类或父类。

成本效益 (cost-effectiveness) 是在开发、维护

和运行信息系统的成本与从系统中获得的效益之间取得平衡。成本效益使用一种称为成本效益分析的技术。

持续过程改进 (Continuous Process Improvement, CPI) 是连续地监控业务过程对降低成本和增加效益方面虽微小但可度量的改善之影响。

持续类 (persistent class) 是描述存活期超过创建它的程序执行时间的对象类。提供读写数据库中的持续属性的功能的对象类。

抽象用例 (abstract use case) 通过组合几个用例中公共的步骤来降低用例之间的冗余。

抽样 (sampling) 是收集文档、表和记录中有代表性的样本的过程。

初学者用户 (novice user) 是不太有经验的计算机用户。

触发器 (trigger) 是嵌入在表中的程序，当修改另一个表时，它就被自动地调用。

次键 (secondary key) 是其值标识一个记录或者所有记录的一个子集的字段。

存储过程 (stored procedures) 是嵌入在表中的程序，可以从一个应用程序调用。

代理 (agent) 是可复用的软件对象，可以在不同的应用软件和网络之间运行。

单元或程序测试 (unit or program test) 是对整个程序的测试。

导出属性 (derived attribute) 是其值可以从其他属性中计算出来或者可以从其他属性值通过逻辑导出的属性。

递归关系 (recursive relationship) 是存在于同一个实体的不同实例之间的关系。

第一范式 (First Normal Form, 1NF) 实体的所有属性对于实体的单个实例都只具有一个值。

第二范式 (Second Normal Form, 2NF) 实体的所有非主键属性的值都依赖于主键。

第三范式 (Third Normal Form, 3NF) 实体的非主键属性的值不依赖于任何其他非主键属性。

电子商务 (electronic commerce, e-commerce) 是指通过使用因特网购买和销售商品及服务。

电子数据交换 (Electronic Data Interchange, EDI)

是企业之间业务事务或数据的标准化电子流。

电子商务 (electronic business, e-business) 是指通过使用因特网进行日常的商务活动。

调查表 (questionnaire) 是一种具有特殊目的的文档, 分析员可以使用它从回答者那里收集信息和观点。

调查研究 (fact-finding) 是一个正式过程, 使用调研、面谈、会议、调查表、抽样和其他技术收集关于系统、需求和喜好的信息。这个活动也称为信息收集或者数据收集。

度数 (degree) 是参与那个关系的实体数量。

对话 (dialogue) 是屏幕和消息的整个流程。

对象 (object) 封装了描述离散的个人、对象、地点、事件或事物的数据 (称为属性) 以及所有使用或修改数据和属性的过程 (称为方法)。访问或修改对象的数据的唯一方法是使用对象预定义的过程。

对象/类关系 (object class relationship) 是一种存在于一个或多个对象/类之间的自然业务联系。

对象技术 (object technology) 是一种软件技术, 它采用封装了数据和行为的对象来定义系统。对于软件开发人员来说, 对象是可复用和可扩展的。

对象建模 (object modeling) 是一种用于辨识系统环境中的对象和这些对象之间关系的技术。

对象建模技术 (object modeling) 试图在被称为对象的单一结构中融合数据和过程要素。对象模型从对象和对象之间关系的角度文档化系统。对象建模技术是面向对象分析和设计方法的基础。

对象框架 (object framework) 是一套互相作用的相关对象, 它们为实现某个任务提供了一套定义良好的服务。

对象实例 (object instance) 由描述特定的人、地点、事物或者事件的属性值构成。

对象责任 (object responsibility) 是当被请求时对象必须提供服务的义务, 如果需要, 它应该与其他对象协作以满足请求。

对象状态 (object state) 是对象在其生命期中某一点所处的条件。

多重性 (multiplicity) 定义一个对象/类对应相关对象/类的一个实例关联可能的最小出现次数和最大出现次数。

多态性 (polymorphism) 是指“多种形式”, 意

味着不同的对象可以以不同的形式响应同样的消息。

反向调度 (reverse scheduling) 建立项目的最后期限, 然后从这个日期开始向后安排进度。

范围蔓延 (scope creep) 是一种常见现象, 其中项目的需求和预期经常不顾对预算和进度的影响而缓慢增加。

方法 (method) 是响应一条消息而执行的软件逻辑。

访问完整性 (referential integrity) 确保一个表中的一个外键值匹配相关表中的主键值。

非功能需求 (nonfunctional requirement) 描述一个满意的系统的其他特征、特点和约束条件。

非结构化面谈 (unstructured interview) 仅仅用一个头脑中的一般目标或目的以及几个 (如果有的话) 特定的问题进行组织。接见者指望被接见者提供一个框架并引导谈话过程。

非确定性关系 (nonidentifying relationship) 是每个参与关系的实体都有各自的独立主键的关系。

非特定关系 (nonspecific relationship) 是一个实体的多个实例同另一个实体的多个实例相关联的关系, 也称为多对多关系。

分布式表现 (distributed presentation) 客户/服务器系统的表现层和表现逻辑层被从遗留系统的服务器上移到客户端上。

分布式关系数据库管理系统 (distributed relational database management system) 是实现分布式关系数据库的软件。

分布式数据 (distributed data) 客户/服务器系统中, 数据层和数据处理层放置在服务器上, 而应用逻辑层、表现逻辑层和表现层放置在客户端。这也称为两层客户/服务器计算。

分布式数据和应用 (distributed data and application) 客户/服务器系统中, 数据层和数据处理层放置在各自的服务器上, 应用逻辑层放置在各自的服务器上, 表现逻辑层和表现层放置在客户端上。这也称为三层或 n 层客户/服务器计算。

分布式系统 (distributed system) 是一个系统, 其构件分布在计算机网络和多个地点。

分层抽样 (stratification) 是一种系统化的抽样技术, 它试图通过将抽样分散 (例如, 按照公式选择文档或记录) 并通过避免过高或过低的估计来减少估计方差。

分割 (partitioning) 是确定如何在网络中最优地

分布或复制应用构件的行为。

分解 (decomposition) 是将一个系统分解成子系统的行为。

分解图 (decomposition diagram) 是一种用来描述系统分解的工具,也称为层次图。

分块因子 (blocking factor) 是包含在一个读或写操作中的逻辑记录数。

分区输出 (zoned output) 是把文本和数字放置在一个表格或屏幕的指定区域或方框中的输出。

分析瘫痪症 (analysis paralysis) 是一个讽刺词汇,描述了一种常见的项目状态,这时过多的系统建模极大地减缓了实现系统方案的进度。

分页显示 (paging) 一次显示一满屏字符。

分支的数据流 (diverging data flow) 是一个分成多个数据流的数据流。

风险管理 (risk management) 是及早发现项目中可能出现的错误,防止它真正威胁信息系统的实现,对它们进行评估和控制。风险管理由风险分析和风险评估所驱动。

封闭式问题 (closed-ended question) 把回答严格限制在特定的选择范围内,或者限制为简短的直接回答。

封装 (encapsulation) 是几项内容一起打包成一个单元(也称为信息隐藏)。

复合键 (concatenated key) 是唯一地标识实体的一个实例的一组属性。同义词包括组合键和合成键。

富余时间 (slack time) 是一个任务的开始时间和结束时间之间可以忍受的延迟量,这个延迟量不会引起整个项目完成时间上的延误。

泛化 (generalization) 是指将几类实体公共的属性组合成独立的实体。

泛化/特化 (generalization/specialization) 是一种技术,其中几类对象类的公共属性和行为被组合成类,称为超类。超类的属性和方法然后被那些对象类(子类)继承。

甘特图 (gantt chart) 是一种条形图,以日历为基准描述项目任务。

工作陈述 (statement of work) 是与管理层和用户团体之间的开发或提升某个信息系统的合约,它定义目标、范围、约束条件、高级用户需求、进度和预算。同义词包括项目章程、项目计划和服务等级合约。

工作抽样 (work sampling) 是一种调查研究技术,它包括以随机间隔进行的大量观察。

工作分解结构 (Work Breakdown Structure, WBS)

是一种图形化工具,用来描述项目分解成开发阶段、开发活动和开发任务的层次结构。

功能 (function) 是企业的一套相关的和正在进行的活动的。

功能键 (function key) 是用于编程特殊操作的一组按键。

功能需求 (functional requirement) 描述一个系统必须提供的活动和服务。

供应链管理 (supply chain management, SCM)

是一种应用软件,它通过直接将企业的信息系统与企业的供应商和分销商的信息系统集成,优化从原材料采购到最终产品分销的业务过程。

估计 (estimation) 是对系统开发所需的费用和努力的预测值。一个不太恰当的同义词是猜测,意思是估计基于实验或试验性证据,但缺少严密性——换句话说,猜测。

固定成本 (fixed cost) 是有规则的但相对固定的费用。

固定格式调查表 (fixed-format questionnaire)

由需要从预先定义的答案中做出选择的问题构成。

关键路径 (critical path) 是一个相关任务序列,该路径决定了项目最早可能完成的时间。

关联关系 (association) 是一个参与者与一个用例发生交互的关系。

关联人员 (stakeholder) 是与某个已存在的信息系统或新信息系统有利益关系的人。关联人员可以是技术工作者,也可以是非技术工作者;既包括内部工作人员,也包括外部工作人员。

关联实体 (associative entity) 是一个从多个其他实体继承其主键的实体。

关系数据库 (relational database) 是一种数据库,它在一系列二维表中存储数据,这些表通过外键互相“关联”。

观察 (observation) 是一种调查研究技术,系统分析员或者参与到活动中,或者观察他人执行活动来了解系统。

管理信息系统 (Management Information System, MIS) 是一种提供面向管理的企业业务处理和运作报告的信息系统。

归档文件 (archival file) 是包含了已经从联机存储中删除了的主文件记录和事务文件记录的表。

规程 (procedure) 是实现一个业务过程的按部

就班的指令和逻辑。

规范化 (normalization) 是一种数据分析技术, 该技术组织数据属性以便它们可以组合起来形成无冗余的、稳定的、灵活的并具有适应性的实体。

滚动显示 (scrolling) 在屏幕中向上或向下移动显示信息, 一次移动一行。

过程 (process) 是在输入数据流或条件上执行, 或者对输入数据流或条件做出响应的工作, 同义词是转换。

过程管理 (process management) 是指定义、改进和协调一个组织为所有系统开发项目所选的系统开发方法 (开发过程), 过程管理关心项目的阶段、活动、交付产品和质量标准, 一致地应用于所有项目。

过程管理软件 (process manager application) 是一个自动化工具, 它帮助记录文档与管理一套方法学和开发路线、交付成果以及质量管理标准。它的同义词是 methodware。

过程建模 (process modeling) 是一种以过程为中心的技术, 其最流行的形式是结构化分析和设计方法, 使用描述业务过程需求的模型推导出有效的软件设计。结构化分析引入了一个称为数据流图的建模工具, 用来说明数据通过一系列业务过程的流程。结构化设计将数据流图转变成一个称为结构图的过程模型, 说明实现业务需求的自顶向下的软件结构。

过程-位置-关联矩阵 (process-to-location-association matrix) 是用来记录过程及其在哪个地点执行的表格。

过程需求 (process requirement) 是关于某个业务过程及其信息系统的处理需求的用户理解。

合并的数据流 (converging data flow) 是多个数据流合并成一个数据流后的数据流。

合成 (composition) 是一种聚合关系, 其中“整体”负责其“部分”的创建和销毁, 如果“整体”不存在了, “部分”也将不存在。

后端信息系统 (back-office information system) 是指支持组织内部业务运行并直达供应商的信息系统。

候选键 (candidate key) 是一组可以作为一个实体的主键的键。有时称为候选标识符。

候选系统矩阵 (candidate systems matrix) 是用来记录候选系统之间异同点的工具。

回转输出 (turnaround output) 是指可能重新进入系统作为输入的外部输出。

活动图 (activity diagram) 是用来图形化地描述业务过程、用例的步骤和对象行为 (方法) 的逻辑的流程的图形。

获取原型 (discovery prototyping) 向用户提供响应需求的一个快速而粗略的实现, 以确定用户的业务需求。

机会 (opportunity) 是指即使在没有出现具体问题的情况下, 也能改善组织的可能性。

积压项目 (backlog) 是由于优先级低于被批准的项目而无法获得经费支持或人员支持的项目建议。注意优先级是不断变化的, 所以, 积压项目在以后也可能会被批准。

基本过程 (elementary process) 是为完成一个事件的响应所需要的离散的详细活动或任务, 也称为原子过程。

基数 (cardinality) 定义了一个实体相对于另一个关联实体的某个具体值的最小和最大具体值数量。

集体讨论 (brainstorming) 是一种用于在小组会议期间产生想法的技术。参与者被鼓励在短时间内产生尽可能多的想法, 而且在所有的想法都提出之前不进行任何分析。

集中式系统 (centralized system) 是一个系统, 其所有构件都在一个集中的多用户的计算机中。

计算机辅助软件工程 (Computer-Assisted Software Engineering, CASE) 使用支持系统模型的绘制和分析的自动化工具, 有些 CASE 工具也提供原型设计和代码生成能力。

记录 (record) 是按照预定义格式安排的字段集合。

记录文档 (documentation) 是记录一个系统的事实和规格说明以供现在和以后参考的活动。

技术可行性 (technical feasibility) 是对一种技术方案的现实性以及技术资源和专家的可用性的度量。

继承 (inheritance) 是指在一个对象类中定义的方法和/或属性可以被另一个对象类继承或复用。

建议申报书 (request for proposal, RFP) 是一种与软件供应商交流应用软件包的业务、技术和支持需求的正式文档, 这些软件供应商希望竞争销售应用软件包或服务。

键 (key) 是一个属性 (或一组属性), 它们对每个实体实例具有一个唯一的值。有时也称为标识符。

交叉功能的信息系统 (cross-functional information system) 支持来自几个业务功能的相关业务过程, 而不考虑传统的组织边界, 例如分部、部门、中心和办事处。

角色扮演 (role playing) 是通过扮演一个对象的行为和责任模拟对象行为和协作的行动。

角色名称 (role name) 是外键的名字, 反映了外键在表中的用途。

接口类 (interface class) 是提供参与者与系统交互的对象。例如: 窗口、对话框或者屏幕。对于非人类参与者, 应用编程接口 (API) 就是接口类, 有时称为界类。

接口说明 (interface specifications) 是记录系统用户如何同系统交互, 以及系统如何同其他系统交互的技术设计。

结构化分析 (structured analysis) 是模型驱动的、以过程为中心的技术, 用于分析一个现有系统, 定义新系统的业务需求, 或者同时用于这两种用途。模型是展示系统组件的图形, 内容包括过程及其相关的输入、输出和文件。

结构化面谈 (structured interview) 在面议中, 接见者有一套专门的问题询问被接见者。

进度可行性 (schedule feasibility) 是对项目时间表的合理性的度量。

经济可行性 (economic feasibility) 是对一个项目或方案的成本效益的度量。

净现值 (net present value) 是一种比较不同方案的年度贴现成本和收益的分析技术。

局域网 (Local Area Network, LAN) 是一组客户端计算机在相对短的距离内连接到一个或多个服务器。

聚合 (aggregation) 是一种关系, 其中一个较大的“整体”类包含一个或多个较小的“部分”类。相反, 一个较小的“部分”类是一个较大的“整体”类的一部分。

决策表 (decision table) 是一张表格, 说明了一组条件及其对应的行动。

决策支持系统 (Decision support system, DSS) 是一种信息系统, 辅助制定决策, 或者提供制定决策的信息。

开放式问题 (open-ended question) 允许被接见者以任何认为是合适的方式回答。

可见性 (visibility) 是外部对象对某个属性或方法的访问等级。

可行性 (feasibility) 是对组织将要开发的信息系统的价值或实用性的度量。

可行性分析 (feasibility analysis) 是度量和评估可行性的活动。

可行性分析矩阵 (feasibility analysis matrix) 是用来评定候选系统的工具。

可用性分析 (usability analysis) 是对系统用户界面的测试。

客户/服务器系统 (client server system) 是一种分布式计算方案, 其中表现层、表现逻辑层、应用逻辑层、数据处理层和数据层分布在客户端 PC 和一个或多个服务器之间。

客户关系管理 (customer relationship management, CRM) 是一种应用软件, 它为客户提供对企业过程的访问能力, 从初始的咨询, 直到售后服务和支持。

空间关系学 (proxemics) 是人与围绕其空间之间的关系的学问。

控制类 (control class) 是承载了不属于实体对象责任的应用逻辑的对象。这种逻辑如: 涉及多个实体对象类的业务规则和计算。控制类协调接口类和实体类之间的消息, 以及消息发送的顺序。

控制流 (control flow) 表示触发一个过程的条件或非数据事件。

跨生命周期活动 (cross life-cycle activity) 是存在于系统开发方法中多个阶段或者所有阶段的活动。例如: 调查研究、记录文档、演示汇报、估计和度量、可行性分析、项目和过程管理、变化管理及质量管理。

快速架构分析 (rapid architected analysis) 也试图从现有系统或获取原型中导出系统模型。

快速应用开发 (Rapid Application Development, RAD) 是一种系统开发策略, 该策略强调用户深入地参与到一系列系统工作原型的快速进化和构造过程中, 以加速系统的开发过程, 系统工作原型最终将成为目标系统 (或者系统的一个版本)。

扩展用例 (extension use case) 是一个由从某个更复杂的用例中提取出来的步骤构成的用例, 以便简化原始用例并扩展其功能。

类图 (class diagram) 是系统静态对象结构的图形描述, 显示了构成系统的对象类, 以及这些对象类之间的关系。

里程碑 (milestone) 是标识项目主要交付成果的完成的事件。

例外报告 (exception report) 是过滤数据以报告对某些情况或标准的例外信息的内部输出。

联合项目计划 (Joint Project Planning, JPP) 是一种策略, 其中项目的所有关联人员参加一个研讨会, 就项目决策达成一致意见。

联合需求计划 (Joint Requirements Planning, JRP) 通过研讨会将所有的系统所有者、系统用户、系统分析员和一些系统设计人员及构造人员组织在一起, 进行系统分析。JRP 一般被看作是联合应用开发 (JAD) 的一部分, JAD 是一种更全面的应用 JRP 于整个系统开发过程的技术。

联机处理 (online processing) 是一种数据处理方法, 其中事务的数据立即处理。

临时对象类 (transient object class) 是描述由程序临时创建并只在程序执行期间存在的对象的类。

逻辑模型 (logical model) 是描述系统是什么或者系统做什么的非技术性的图形化表示。同义词包括本质模型、概念模型和业务模型。

逻辑设计 (logical design) 将业务用户需求转换成系统模型, 该模型仅仅描绘了业务需求, 而没有描述这些需求的任何可能的技术设计或实现。常见的同义词有: 概念设计和要点设计。后者指的是建模系统的“要点”, 或者独立于任何技术的“基本需求”。逻辑设计的反义词是物理设计 (在本章后面定义)。

面谈 (interview) 是一种调查研究技术, 系统分析员借此通过面对面的交互从个人那里收集信息。

面向对象分析 (Object-Oriented Analysis, OOA)

1) 研究现有对象, 看它们是否能够被复用或者被调整用于新的用途; 2) 定义各种新对象和修改后的对象, 它们将与现有对象一起组合成一个有用的企业计算应用系统。

面向对象方法 (Object-Oriented Approach, OOA)

是一种模型驱动的技术, 它将数据和过程集成到被称为对象的结构中。对象模型是从各个方面 (例如结构和行为以及对象的交互) 说明系统的对象的图形。

面向对象分析和设计 (object-oriented analysis and design, OOAD) 是用于系统开发的一组工具和技术的集合, 利用对象技术来构造系统及其软件。

面向对象设计 (Object-oriented design, OOD)

采用协作的对象、对象的属性和方法说明软件解决方案的一种方式。

敏捷方法 (agile method) 集成各种系统分析和

设计方法, 根据要解决的问题和要开发的系统应用合适的方法。

敏捷开发 (agile development) 是一种系统开发策略, 系统开发人员可以从一套相应的工具和技术中灵活地选择最适合完成手边任务的工具和技术。敏捷开发被认为可以在系统开发的产量和质量之间达到最优化的平衡。

模块测试 (stub test) 是对程序的一个子集进行的测试。

模型 (model) 是对现实或构想的一种表述。因为“一幅图胜过于言万语”, 所以大多数模型使用图形方式表述现实或构想。

模型驱动分析 (model-driven analysis) 强调绘制图形化系统模型来记录和验证现有的和/或建议的系统。系统模型最终将成为设计和构造一个改进的系统的蓝图。

模型驱动开发 (Model-Driven Development, MDD) 技术强调绘制模型以可视化地分析问题、定义业务需求以及设计信息系统。

模型驱动设计 (model-driven design) 是一种系统设计方法, 强调通过绘制系统模型记录系统的技术或实现方面。

默认值 (default value) 是如果用户没有指定值的话将被记录的值。

目标 (objective) 是一种对成功的度量。它是假定在有足够资源的条件下希望实现的某些东西。

内部输出 (internal output) 是提供给系统所有者和组织内的系统用户的输出。

内联网 (intranet) 是一个使用因特网技术将桌面、工作组和企业计算集成在一起的服务器网络。

能力成熟度模型 (Capability Maturity Model, CMM) 是用来评估组织的信息系统开发以及管理过程和产品的成熟度等级的框架。它由 5 个开发成熟度等级构成。

逆向工程 (reverse engineering) 是 CASE 工具的一种能力, 能够直接从软件或数据库代码生成初始的系统模型。

逆向工程技术 (reverse engineering) 读取一个现有数据库、应用程序和/或用户界面的程序代码, 并自动地生成等价的系统模型。

胖客户 (fat client) 是一台功能更强大的个人计算机、笔记本计算机或工作站。

批处理 (batch processing) 是一种数据处理方法, 其中许多事务的数据被收集到一个文件中

进行处理。

平衡 (balancing) 要求不同详细程度的数据流程图保持一致性和完整性。

期望工期 (expected duration, ED) 估计完成任务所需的时间量。

企业应用集成 (enterprise application integration, EAI) 是指用来链接应用软件以支持应用软件之间的数据和信息流的过程和技术。EAI 解决方案通常基于中间件。

企业战略规划 (strategic enterprise plan) 是整个企业的战略规划 (一般 3 年到 5 年), 定义企业的任务、愿景、目标、战略、基准和评价准则。企业战略规划一般由业务部门战略规划补充完善, 定义每个部门如何为企业做出贡献。(以上的) 信息系统战略规划是部门级规划之一。

企业资源规划 (Enterprise Resource Planning, ERP) 是一种应用软件, 它将信息系统完全集成在一起, 提供大部分或者所有核心基本业务功能 (包括这些业务功能所需的事务处理和管理信息)。

前端信息系统 (front-office information system) 是指支持延伸到企业客户的业务功能的信息系统。

全面质量管理 (Total Quality Management, TQM) 是一种在企业内部促进质量改善和管理的综合方法。

确定性关系 (identifying relationship) 是父实体贡献其主键成为子实体的主键的一部分的关系。

软件尺度 (software metrics) 是对软件质量和生产率的数学度量。

软件规格说明 (software specification) 是业务过程的技术设计, 这些业务过程将由系统构造人员编写的计算机程序自动化执行或者提供支持。

软件开发环境 (Software Development Environment, SDE) 是用于构造信息系统应用的语言和工具包。

商用应用软件包 (commercial application package) 是一种可以购买到并 (在一定限度内) 定制的软件应用, 以满足大量组织或特定行业的业务需求。同义词是商用现成产品 (COTS) 系统。

上下文数据流图 (context data flow diagram) 是用来记录系统的范围的过程模型, 也称为环境

模型。

设计单元 (design unit) 是一个自包含的过程、数据存储和数据流集合, 它们共享了类似的设计属性。

设计类图 (design class diagram) 是一个图形, 描述与用于构建程序的软件组件相对应的类。

设计模式 (design pattern) 是对一个给定上下文中一个给定问题的通用方案, 它支持了对已证明的方法和技术的复用。

审计测试 (audit test) 是确保系统准备就绪可以运行的测试。

审计文件 (audit file) 是包含了修改其他文件的记录的表。

时间盒 (timeboxing) 是一段不能延长的时间段 (通常为 60 ~ 90 天), 系统的第一个版本 (或下一个版本) 必须在这个时间段内投入运行。

时间盒技术 (timeboxing) 是一种以版本化的形式发布信息系统功能和需求的技术。开发团队选择系统的最小子集, 这个子集如果完全实现, 就能够立即向系统所有者和用户返回价值。理想情况下, 那个子集应该在 6 ~ 9 个月或更少的时间内开发出来; 之后, 系统的增值版本以类似的时间段开发出来。

时序事件 (temporal event) 是由时间触发的系统事件。

实体 (entity) 是我们需要收集数据和存储数据的人、地点、对象、事件或概念的类。

实体类 (entity class) 是承载业务相关信息 (特别是持续的、存储到数据库中的信息) 的对象类。

实体关系图 (Entity Relationship Diagram, ERD) 是一种利用符号记法按照数据描述的实体和关系来刻画数据的数据模型。

实体实例 (entity instance) 是实体的具体值。

事件 (event) 是必须作为一个整体完成的逻辑单位工作, 有时称为事务。

事件图 (event diagram) 是描述一个事件的上下文图的数据流图。

事务处理系统 (Transaction Processing System, TPS) 是一种捕捉和处理有关企业事务数据的信息系统。

事务服务器 (transaction server) 是运行确保所有数据库修改作为一个整体成功或者失败的服务的服务器。

事务文件 (transaction file) 是包含了描述业务事件的记录的表。

瘦客户 (thin client) 是一台功能不十分强大的个人计算机。

鼠标 (mouse) 是一种引起指针在屏幕上移动的设备。

数据 (data) 是组织内部关于人、地点、事件和事务的重要的原始事实。每个事实本身并没有什么意义。

数据仓库 (data warehouse) 是存储从运行数据库中提取的数据的数据库。

数据处理语言 (Data manipulation Language, DML) 是用来创建、读取、修改和删除记录的 DBMS 语言。

数据存储 (data store) 存储数据供日后使用。同义词包括文件和数据库。

数据-地点-CRUD 矩阵 (data-to-location-CRUD matrix) 是用来将数据需求映射到地点的矩阵。

数据定义语言 (data definition language, DDL) 是 DBMS 用来定义数据库或数据库视图的语言。

数据分析 (data analysis) 是为将数据模型实现成数据库而改进数据模型的技术。

数据管理员 (data administrator) 是负责数据规划、数据定义、数据架构和数据管理的数据库专家。

数据架构 (data architecture) 定义如何开发文件和数据库。

数据建模 (data modeling) 是一种以数据为中心的技术, 用来建模业务数据需求, 以及设计实现这些需求的数据库系统。最常见的数据模型是实体关系图。有时称为数据库建模。

数据结构 (data structure) 是数据属性的特定排列, 它定义了数据流实例的一个实例。

数据库 (database) 是相关文件的集合。

数据库服务器 (database server) 是运行一个或多个数据库的服务器。

数据库管理系统 (database management system, DBMS) 是用于创建、访问、控制和管理数据库的专用软件。

数据库管理员 (database administrator) 是负责数据库技术, 数据库设计和构造咨询, 安全、备份和恢复, 以及性能调试的专家。

数据库架构 (database architecture) 是指用于支持数据架构的数据库技术。

数据库模式 (database schema) 是表示数据库的技术实现的模型或蓝图。

数据类型 (data type) 是属性的一个参数, 定义

了这个属性中可以存储什么类型的数据。

数据流 (data flow) 是一个过程的数据输入, 或者来自一个过程的数据 (或信息) 输出。

数据流图 (data flow diagram, DFD) 是一种描述通过系统的数据流以及系统实施的工作或处理过程的过程模型。同义词包括泡式图、转换图和过程模型。

数据录入 (data entry) 是把数据翻译成计算机可读格式的过程。

数据收集 (data capture) 是新数据的标识和获取。

数据守恒 (data conservation) 是确保一个数据流只包含接收数据的过程真正需要的数据的实践。

数据需求 (data requirement) 是用户数据以实体、属性、关系和规则形式的表述。

数据属性 (data attribute) 是对最终用户和业务有意义的最小数据块。

水平分层 (clean layering) 是一种设计策略, 它要求表现层、应用层和数据层被物理地分离。

顺序图 (sequence diagram) 是一种 UML 图, 它通过描述对象按照时间顺序的消息交互来建模用例逻辑。

随机抽样 (randomization) 是一种抽样技术, 其特点是选择样本数据时没有预定的模式或计划。

特征蔓延 (feature creep) 是不受控制地增加技术特征到一个系统中。

替代键 (alternate key) 是没有被选中作为主键的任何候选键。

通信和协作系统 (communication and collaboration system) 促进工作人员、合作伙伴、顾客和供应商之间进行更有效的通信, 以提高他们协作能力的信息系统。

统一建模语言 (Unified Modeling Language) 是一套建模规则, 它使用对象说明或描述软件系统。

投资回报率 (ROI) 分析 (return-on-investment analysis) 是一种比较替代方案或项目的终生收益率的技术。

投资回收分析 (payback analysis) 是一种用于确定投资是否可以收回以及何时收回的技术。

投资回收期 (payback period) 是产生的收益超过产生的成本之前所经历的时间。

图形输出 (graphic output) 是使用图片化图表来揭示信息的输出。

外部代理 (external agent) 是与系统交互的外部

的人员、组织部门、其他系统或者其他组织，也称为外部实体。

外部服务提供者 (External Service Provider, ESP) 是指销售他们的专业知识和经验给其他企业，帮助那些企业购买、开发和集成企业信息系统的系统分析员、系统设计人员或者系统构造人员。他们可能属于某些咨询机构或服务机构。

外键 (foreign key) 是一个实体的主键，它被贡献给（复制到）另一个实体以确定一个关系实例。

网络计算系统 (network computing system) 是一种多层方案，其中表现层和表现逻辑层在客户端 Web 浏览器中使用从某个 Web 服务器下载的内容实现。

文档文件 (document file) 是包含了历史数据的表。

文件 (file) 是相似记录的集合。

问题 (problem) 是不期望发生的情况，它妨碍组织完整地实现其任务、愿景或目标。

问题陈述 (problem statement) 是对问题、机会和指示的描述和分类，也可能包括约束条件和对解决方案的一个初始视图。同义词包括初始研究和可行性评估。

无形收益 (intangible benefit) 是指被认为难以量化或者不可能量化的收益。

物理模型 (physical model) 是展示系统是什么或者系统做什么，以及系统如何实现的技术性的图形化表示。同义词包括实现模型和技术模型。

物理设计 (physical design) 将业务用户需求转换成系统模型，描述用户的业务需求的技术实现。常见同义词包括：技术设计或实现模型。反义词是逻辑设计。

物理数据流图 (physical data flow diagram) 是一个过程模型，用于交流信息系统的技术实现特征。

系统测试 (system test) 是对整个系统的测试。

系统方案建议 (system proposal) 是被推荐系统的一份书面报告或演示汇报。

系统分析 (system analysis) 是研究业务问题领域，以推荐改进措施并说明方案的业务需求和优先权。

系统分析用例 (system analysis use case) 是记录系统用户和系统交互的用例，描述什么需求需要高度细化，但仍没有太多的实现细节和

约束。

系统分析员 (system analyst) 研究组织存在的问题和需求，确定人员、数据、过程和信息技术如何最大化地为企业做出贡献。

系统构造 (systems construction) 是系统组件的开发、安装和测试。

系统构造人员 (system builder) 是根据系统设计人员的设计说明构造信息系统及其构件的技术专家。

系统开发方法 (systems development methodology) 是一个十分正式且精确的系统开发过程，它为系统开发人员和项目管理者定义了（在 CMM 第 3 级）一组活动、方法、最佳实践、交付成果和自动化工具，用来开发和维护大部分或所有的信息系统和软件。它的一个同义词是系统开发过程。

系统开发过程 (system development process) 是一组活动、方法、最佳实践、交付成果和自动化工具，系统开发的关联人员用它们来开发和维护信息系统及软件。

系统类 (system class) 是处理操作系统相关功能的对象类。

系统模型 (system model) 是系统的一幅图示，表示了现实情况或者希望的情况。系统模型促进了系统用户、系统分析员、系统设计人员和系统构造人员之间的交流。

系统启动 (system initiation) 是项目的初始规划，定义初始业务范围、目标、进度和预算。

系统设计 (system design) 为系统分析阶段确定的业务需求设计（或构造）一个技术性的基于计算机的方案。（注意：越来越多的设计采用原型系统的形式。）

系统设计人员 (system designer) 是将系统用户的业务需求和约束条件转换成技术方案的技术专家。他们设计满足系统用户需求的计算机数据库、输入、输出、屏幕界面、网络和程序。

系统生命周期 (system life cycle) 将一个信息系统的生命分为两个阶段：1) 系统开发阶段；2) 系统运行和支持阶段，首先构建系统；然后使用系统，运行系统并支持系统；最后，从运行和支持阶段再回到开发阶段。

系统实现 (system implementation) 构造、安装、测试和发布一个系统投入生产（即日常运行）的过程。

系统思想 (system thinking) 是形式化的系统理论和概念在系统问题解决中的应用。

系统所有者 (system owner) 是信息系统的发起人和主要倡导者,他们通常在项目的信息系统开发、运行和维护上提供资金。

系统需求 (system requirement) 是信息系统必须实现的或者必须具备的属性。

系统验收测试 (systems acceptance test) 是对最终系统进行的测试,其中用户进行验证测试、确认测试和审计测试。

系统用户 (system owner) 是那些在通常意义上使用信息系统或者受到信息系统影响的“客户”——如收集、验证、录入、响应、存储、交换数据和信息。

系统支持 (system support) 是对系统用户的不断的技术支持,以及处理可能出现的错误、失误或新的需求所需的维护工作。

现代结构化设计 (modern structured design) 是一种系统设计技术,它将系统过程分解成可管理的构件。

现值 (present value) 是在未来任何时候 1 美元的当前价值。

详细报告 (detailed report) 是显示很少 (或者没有) 经过过滤的信息的内部输出。

项目 (project) 是必须按时在预算内并遵循规格说明完成的一系列活动。

项目范围 (project scope) 定义项目的边界——项目可能 (或者可能不) 涉及的业务领域。

项目管理 (project management) 是界定范围、规划、组织人员、组织、指导和控制一个项目的活动,以最小成本、在规定时间内,以可接受的质量开发信息系统。

项目管理 (project management) 是指为了在指定的时间和预算范围内开发出一个可接受的系统而定义、规划、指导、监视和控制项目的活动。

项目管理软件 (project manager application) 是一个自动化工具,它帮助规划系统开发活动 (最好使用认可的方法学)、估计和分配资源 (包括人力和经费)、调度活动和资源、按照进度和预算监督进展、控制和修改进度和资源,以及报告项目进展。

项目经理 (project manager) 是经验丰富的从业人员,主要负责根据进度安排、预算、发布的产品、客户满意度、技术标准和系统质量,计划、监督和控制项目。

消息 (message) 是当一个对象调用另一个对象的方法 (行为) 以请求信息或者某些动作时发

生的通信。

消息或组件服务器 (messaging or groupware server) 是运行组件服务的服务器。

协作图 (collaboration diagram) 是一种 UML 图,它通过描述对象按照消息顺序的消息交互来建模用例逻辑。

信息 (information) 是为某些人进行了处理或重新组织成更有意义的形式的数据。信息通过数据的组合形成,这种组合期望对接收者有意义。

信息工程 (information engineering, IE) 是一种用来计划、分析和设计信息系统的模型驱动的、以数据为中心的但对过程敏感的技术。IE 模型是一些说明和同步系统的数据和过程的图形。

信息工作者 (information worker) 指在工作中涉及到创建、收集、处理、分发和使用信息的人。

信息技术 (Information Technology, IT) 是一个现代词汇,描述了计算机技术 (硬件和软件) 和电信技术 (数据、图像和语音网络) 的组合。

信息系统 (Information System, IS) 是人、数据、过程和信息技术组合,它们之间相互作用,收集、处理、存储和提供支持企业运作的信息。

信息系统分析 (information systems analysis) 定义为一个信息系统开发项目中的这样一些开发阶段,这些阶段的重点是业务问题和需求,这些需求独立于实现方案中可能使用的任何技术。

信息系统架构 (information systems architecture) 提供一个统一的框架,在这个框架中,各种具有不同观点的人可以组织并查阅信息系统的基本构件。

信息系统战略规划 (strategic information systems plan) 是一个正式的战略规划 (一般 3 年到 5 年),用于构造和改进信息技术架构以及使用此架构的信息系统。

行为 (behavior) 是指对象可以做的事情,以及在对象数据 (或属性) 上执行的功能。在面向对象环境中,对象的行为通常被称为方法、操作或者服务 (我们可能在讨论中交叉地使用这些词汇)。

需求管理 (requirements management) 是管理需求的变化过程。

需求获取 (requirements discovery) 包括系统分

析员用来从用户团体那里确定或提取系统问题和方案需求的那些技术。

演示汇报 (presentation) 是交流发现的情况、建议和文档的活动,供感兴趣的用户和管理者评审。演示汇报可以是书面的,也可以是口头的。

业务功能 (business function) 是支持业务的相关过程的集合。功能可以分解成其他的子功能,最终分解成执行特定任务的过程。

业务过程重构 (Business Process Redesign, BPR) 是指研究、分析和重新设计企业的基本业务过程,为企业降低成本和/或提高效率。

移动用户 (mobile user) 是指位置经常变化,但需要从任意地点都能访问到信息系统的用户。

以用户为中心的开发 (user-centered development) 是一个系统开发过程,该过程基于对关联人员的需求,以及开发该系统的原因的充分理解之上。

因果分析 (cause-and-effect analysis) 是一种研究问题以确定其原因和结果的技术。

应用程序 (application program) 是基于程序设计语言的可以被机器读取的表示,它表示了软件过程要做什么,或者软件过程要如何实现它的任务。

应用服务器 (application server) 是运行信息系统的逻辑和服务的服务器。

应用架构 (application architecture) 是用于实现信息系统的技术规范。

应用开发环境 (Application Development Environment, ADE) 是集成的软件开发工具,它提供了以最快速度和最高质量开发新应用程序所需的全部工具。常用的同义词有集成开发环境 (IDE)。

用户会话 (user dialogue) 描述了用户如何从一个窗口移动到另一个窗口,如何从一个页面移动到另一个页面,如何同应用程序交互、完成有用的工作。

用例 (use case) 是业务场景或事件,系统必须对这些场景或事件提供确定的响应。用例来自面向对象分析中,但它们在许多系统分析和设计方法中都很常见。

用例分级和评估矩阵 (use-case ranking and priority matrix) 是用来评估用例决定其优先级的工具。

用例建模 (use-case modeling) 是使用业务事件、发起业务事件的人,以及系统如何响应这些事件来建模系统功能的过程。

用例描述 (use-case narrative) 是业务事件以及

用户如何同系统交互以完成任务的文字描述。

用例图 (use-case diagram) 是描述系统与外部其他系统以及用户之间交互的图形。换句话说,用例图描述了谁将使用系统,用户希望以什么方式与系统交互。

用例依赖关系图 (use-case dependency diagram) 是用例之间的依赖关系的图形化表述。

有形收益 (tangible benefit) 是那些容易被量化的收益。

鱼骨图 (Ishikawa diagram) 是一种用于确定、探索和描述问题及其原因和结果的图形工具,它经常被称为因果图 (cause-and-effect diagram) 或鱼骨图 (fishbone diagram) (因为它像一个鱼骨)。

预期管理矩阵 (change management matrix) 是一个理解改变项目参数的动态和影响的工具。

域 (domain) 是属性的一个参数,定义了这个属性可以取的合法值。

原型 (prototype) 是一个小规模的、有代表性的或者可工作的模型,这个模型反映了信息系统的用户需求或建议设计。任何原型都可能会忽略某些功能或特征,直到原型最终完全进化成需求的一个可接受的实现系统为止。

原型设计 (prototyping) 是一种使用快速应用开发工具快速构造信息系统的可工作的(但不完整的)模型的技术。

源文档 (source document) 是用来记录业务事务的表格。

远程批处理 (remote batch processing) 是一种数据处理方法,其中数据联机输入,收集成一批,以后再处理。

远程用户 (remote user) 是指不在公司办公地点但仍需要访问信息系统的用户。

约束条件 (constraint) 是将限制你灵活地为目标定义方案的某些东西。约束条件基本上无法被改变。

约束条件 (constraint) 是可能制约解决方案或问题解决过程的因素、限制条件或约束。

运行可行性 (operational feasibility) 是对方案在组织中的合适(或接受)能力的度量。

运行数据库 (operational database) 支持信息系统的日常运行和业务事务处理,也称为事务数据库。

正式汇报 (formal presentation) 是用来兜售新想法并获得新系统认可的专门会议。

正向调度 (forward scheduling) 建立项目开始日

期,然后从这个日期开始向前安排进度。

正向工程 (forward engineering) 是 CASE 工具的一种能力,能够直接从系统模型生成初始软件或数据库代码。

知识 (knowledge) 是依据接收者的事实、真理、信仰、判断、经验和专业知识进一步提炼后的数据和信息。理想情况下,信息产生智慧。

知识工作者 (knowledge worker) 是指其工作基于专业化知识的工作者。

肢体语言 (body language) 是沟通中的非口头信息。

指导部门 (steering body) 是一个由主要业务管理人员和系统管理人员构成的委员会,它研究相互竞争的项目建议,给项目排列先后次序,确定哪个项目将给组织带来最大的价值并因此应该被批准继续进行系统开发。它也称为指导委员会 (steering committee)。

指导委员会 (steering committee) 是由系统所有者和信息技术主管组成的管理机构,它给候选的系统开发项目排序,批准相应项目。

指示 (directive) 是一个由管理层、政府或其他外部影响强加的新需求。

智能键 (intelligent key) 是一个业务编码,其结构表达了一个实体实例的数据。

中间件 (middleware) 用来在不同应用之间转换和路由数据的 (通常是购买的) 软件。

重载 (override) 是一种技术,其中子类使用它自己的属性或行为,而非从父类继承的属性或行为。

逐步投入 (creeping commitment) 是在整个项目过程中都持续地重新评价可行性和风险,并相应地调整项目预算和最后期限。

主管信息系统 (Executive information system, EIS) 是支持主管经理的规划和评估需求的信息系统。

主键 (primary key) 是最常用来唯一地确定一个实体实例的候选键。

主文件 (master file) 是包含了相对稳定的记录的表。

属性 (attribute) 是实体的描述性性质或特征。同义词包括要素、性质和域。

专家系统 (expert system) 捕捉专业技术人员的专业知识,然后模拟这些专业知识为非专家服务的信息系统。

专家用户 (expert user) 是有经验的计算机用户。

状态图 (statechart diagram) 是一种 UML 图,

它描述了一个对象在其生命期中可能的状态组合,触发状态转换的事件,以及决定状态转换的规则。

状态转换事件 (state transition event) 通过修改对象属性的一个或多个值触发对象状态变化的事件。

状态转换图 (state transition diagram, STD) 是一种用于描述用户会话期间可能出现的屏幕的顺序和变化的工具。

资料库 (repository) 是一个数据库和/或文件目录,系统开发者在其中存储一个或多个信息系统或项目的所有文档、知识和产品。知识库通常是自动化的,以便于信息存储、访问和共享。

资源调配 (resource leveling) 是一种改正资源过度分配问题的策略。

子集准则 (subsetting criteria) 是一个属性 (或合成属性),其有限的取值范围把所有的实体实例分成了有用的子集。这有时也称为反向条目。

子类 (subtype) 是一个对象类,它从一个超类继承属性和行为,并可能包含自身所特有的属性和行为,也称为儿子类。如果它位于继承层次的最底层,也称为实类。

字段 (field) 是存储在文件或数据库中的有意义数据的最小单元。

自由格式调查表 (free-format questionnaire) 为回答者提供了很大的回答范围。它提出一个问题,然后回答者在这个问题后面提供的空白区里填写答案。

总结报告 (summary report) 是为管理人员分类信息的内部输出。

组合数据流 (composite data flow) 是由其他数据流构成的数据流。

组合属性 (compound attribute) 实际上是由其他属性构成的属性。它在不同的数据建模语言中有很多同义词:串联属性、合成属性和数据结构。

组件 (component) 是封装在一个单元内的一组对象。组件的一个例子是动态链接库 (DLL) 或者可执行文件。

最差工期 (pessimistic duration, PD) 估计完成任务所需的最大时间量。

最可能工期 (most likely duration, D) 基于最优、最差和期望工期的权重计算完成任务所需的估计时间量。

最优工期 (optimistic duration, OD) 估计完成任务所需的最小时间量。